SIXTH FRAMEWORK PROGRAMME
PRIORITY 2
INFORMATION SOCIETY TECHNOLOGIES

Contract for:

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Annex I – "Description of Work"

Project acronym: SELFMAN
Project full title: Self Management for Large-Scale Distributed Systems based on Structured
Overlay Networks and Components
Proposal/Contract no.: 34084
Related to other Contract no.:

Date of preparation of Annex I: March 29, 2006

Operative commencement date of contract: June 1, 2006

Date of this revision: February 11, 2009

Table of Contents

1. Project Summary

The goal of SELFMAN is to make large-scale distributed applications that are *self managing*, by combining the strong points of component models and structured overlay networks. One of the key obstacles to deploying large-scale applications running on networks such as the Internet or company intranets is the issue of management. Currently many specialized personnel are needed to keep large Internet applications running. SELFMAN will contribute to removing this obstacle, and thus enable the development of many more Internet applications and Internet-based companies that depend on such applications. In the context of SELFMAN, we define self management along four axes: *self configuration* (systems configure themselves according to high-level management policies), *self healing* (systems automatically handle faults and repair them), *self tuning* (systems continuously monitor their performance and adjust their behavior to optimize resource usage and meet service level agreements), and *self protection* (systems protect themselves against security attacks). SELFMAN will provide self management by combining a component model with a structured overlay network. The component model will support dynamic configuration, the ability of part of the system to reconfigure other parts at run-time, which is the key property that underlies the self-management abilities. Basing the system on a structured overlay network enables SELFMAN to extend the abilities of the component model to large-scale distributed systems. Structured overlay networks have made much progress since their origins in peer-to-peer file-sharing applications. In contrast to file-sharing applications, structured overlay networks provide guarantees for efficient communication and reorganization in case of failures. These are already low-level self-management properties. Combining this with the component model, SELFMAN will build high-level self-management properties on top of these low-level properties. SELFMAN will do both foundational research and applied research. The foundational research will design a distributed service architecture that combines structured overlay networks (for communication and basic self management) with component models (for the higher self management primitives). To make the research concrete we will target multi-tier applications, and specifically we will build two-tier applications using a self-managing storage (database) service. We will use industrial trace data to measure the effectiveness of our self managing architecture. We will do implementation work in two directions: first, to explore how an industrial standard platform (J2EE) can be made self-managing, and second, to push self management as far as we can, in terms of fundamental programming language research, without being restrained by existing tools. The interplay between these two implementations will be to the benefit of both. The industrial partners will use the results of SELFMAN to guide their strategic decisions for distributed systems development.

2. Project Objectives

The vision of SELFMAN is that distributed systems should be *self managing*. Self management, as a general concept, means that the system should be able to reconfigure itself to handle changes in its environment or requirements without human intervention but according to high-level management policies. As part of the SELFMAN project, we will give a precise definition of self management that makes it clear what parts can be handled automatically and what parts need application programmer or user (system administrator) intervention. The user then defines a self management policy and the system implements this policy. Self management exists on all levels of the system. At the lowest level, self management means that the system should be able to automatically handle frequent addition or removal of nodes, frequent failure of nodes, load balancing between nodes, and threats from adversaries. For large-scale systems, environmental changes that require some recovery by the system become normal and even frequent events. For example, failure becomes a normal situation: the probability that at a given time instant some part of the system is failed approaches 1 as the number of nodes increases. At higher levels, self management embraces many system properties. For SELFMAN, we consider that these properties are classified in four axes of self management: self configuration, self healing, self tuning, and self protection. An example of self configuration is upgrading part of the system from one version to a later version. Because we expect the system to be a continuously running infrastructure, we require that this versioning can be done without interrupting service.

To be effective, self management must be designed as part of the system from its inception. It is difficult or impossible to add self management a posteriori. This is because self management needs to be done at many levels of the system. Each level of the system needs to provide self management primitives ("hooks") to the next level. This is why SELFMAN makes self management its main goal. The key for enabling self management is the component model. It must have the primitives needed to do all the system monitoring and system modification during its execution.

In the SELFMAN project we will design and build a service architecture that is a framework for building large-scale self-managing distributed applications. The heart of the service architecture is a component model built in synergy with a structured overlay network (see below). The project will start by focusing on the component model and the structured overlay network, which are used together to provide the foundation of the service architecture. We then provide solutions for the four axes of self management by building on this foundation. We consider the most important axis to be self configuration, since it is necessary for the deployment of an application. This gives the following four specific objectives for the self-management abilities:

1. *Self configuration*: We will provide all the necessary configuration primitives at all levels of this service architecture, so that the system can reconfigure itself during execution. Specifically, we will provide primitives so that the service architecture will continue to work when nodes are added or removed during execution. We will provide primitives so that parts of the application can be upgraded without interrupting execution (on-line upgrade) . We will also provide a component trading infrastructure that can be used for automating distributed configuration processes.

2. *Self healing*: The service architecture will provide the primitives for continued execution when nodes fail or when the network communication between nodes fails, and will provide primitives to support the repair of node configurations. Specifically, the service architecture will continue to provide its basic services, namely communication and replicated storage, and will provide resource trading facilities to support repair mechanisms. Other services are application-dependent; the service architecture will provide the primitives to make it easy to write applications that are fault-tolerant and are capable of repairing themselves to continue respecting service level agreements.

3. *Self tuning*: The service architecture will provide the primitives for implementing load balancing and overload management. We expect that both load balancing and on-line upgrade will be supported by the component model, in the form of introspective operations (including the ability to freeze and restart a component and to get/set a component's state).

4. *Self protection*: In a project the size of SELFMAN, it is not possible to provide a general solution to the security problem. That would be the subject of another project. Nevertheless, security is an essential concern that has to be considered. In SELFMAN, we will consider a simple threat model, in which only the nodes of the service architecture are considered trustworthy. We can extend this threat model with little effort for some parts, such as the structured overlay network, for which we already know how to protect against more aggressive threat models.

We will measure the effectiveness of these objectives in three ways. First, by formal reasoning and proofs (e.g., proving correctness of the algorithms as part of the research results). Second, by using industrial usage scenarios and evaluating the qualitative effectiveness with these scenarios. Third, by using industrial trace data and generated traffic, and evaluating the quantitative effectiveness.

An essential feature of self management is that it adds feedback loops throughout the system. A feedback loop consists of (1) the detection of an anomaly, (2) the calculation of a correction, and (3) the application of the correction. These feedback loops exist within one level but can also cross levels. For example, the low level detects a network problem, a higher level is notified and decides to try another communication path, and the low level then implements that decision. The primitives for self management at a given level are of two kinds: *detectors* and *actuators*. Because of the feedback loops, it is important that the system behavior converges (no oscillatory, chaotic, or divergent behavior). One goal of the project is therefore to model formally the feedback loops, to confirm convergent behavior (possibly changing the design), and to validate the model with the system. The formal model of a computer system is generally highly nonlinear. It may be possible to exploit oscillatory or chaotic behavior to enhance certain characteristics of the system. We will explore this aspect of the feedback loops.

3. Participant List

Partic. Role	Partic. No.	Participant name	Participant short name	Country	Date enter project	Date exit project
CO	1	Université catholique de Louvain	UCL	Belgium	Month 1 (start of project)	Month 36 (end of project)
CR	2	Royal Institute of Technology (Kungliga Tekniska Högskolan)	KTH	Sweden	Month 1 (start of project)	Month 36 (end of project)
CR	3	Institut National de Recherche en Informatique et Automatique	INRIA	France	Month 1 (start of project)	Month 36 (end of project)
CR	4	France Telecom Research and Development	FT R&D	France	Month 1 (start of project)	Month 36 (end of project)
CR	5	Konrad-Zuse-Zentrum für Informationstechnik Berlin	ZIB	Germany	Month 1 (start of project)	Month 36 (end of project)
CR	6	E-Plus Mobilfunk GmbH & Co. KG	E-Plus	Germany	Month 1 (start of project)	Month 1
CR	7	National University of Singapore	NUS	Singapore	Month 1 (start of project)	Month 36 (end of project)
CR	8	Stakk AB	Stakk	Sweden	Month 13 (start of 2nd year)	Month 36 (end of project)

4. Relevance to the Objectives of the Specific Programme and/or Thematic Priority

Let us examine the vision of self management, and our specific approach of combining structured overlay networks with a component model, from the viewpoint of the IST 2005-2006 Workprogramme in Software and Services (IST Call 5, Section 2.5.5). This workprogramme has five objectives. We explain how SELFMAN is relevant to these objectives. We list the objectives with the most relevant first.

- The main objective addressed by SELFMAN is *Foundational and applied research to enable the creation of software systems with properties such as self-adaptability, flexibility, robustness, dependability and evolvability.* We provide self-adaptability at the lowest level by using structured overlay networks (which automatically reorganize when nodes are added and removed), and with the component model we will extend it to cover the whole system. The component model addresses the key enabling axis of self management, namely the ability to monitor and modify the structure of the application at run-time. Flexibility and evolvability are made possible by the monitoring and modification abilities of the component model. Robustness and dependability are explicitly addressed by two other axes of self management, namely self healing and self protection.
- The second objective addressed is *Research on the engineering, management and provision of services and software, incorporating ambient intelligence-based features such as dynamic composability and adaptability, context awareness, autonomy and semantic interoperability.* Self management overlaps with context awareness and adaptability: the system detects changes in its environment and must reconfigure itself for them. Self management provides autonomy: for a system to be autonomous, it should be able to maintain itself in a useful working condition despite environmental changes. Note that we do not specifically address dynamic composability and semantic interoperability in this project. However, the results of the project can be applied to these two areas.
- The third objective addressed is *Principles, methodologies and tools for design, management and simulation of complex software systems, viewing the user as part of the system.* In a self-managed system, there are feedback loops at all levels. Some of these feedback loops will contain a human being as part of the loop. In addition, the feedback loops form a complex nonlinear system, in which properties such as convergence, oscillation, and chaotic behavior are important. In SELFMAN we will make a first step toward studying these properties.
- The fourth objective addressed is *Research into technologies specifically supporting the development, deployment, evolution, and benchmarking of open source software.* SELFMAN will do open source software development. The structured overlay network and component model research are relevant for the open source community. Project partners UCL and KTH have more than a decade of experience in open source community activities, through the ongoing development of the Open Source Mozart Programming System. Partners France Telecom and INRIA have experience in open source development through the ObjectWeb consortium.
- The fifth objective addressed is *Support actions contributing to the achievement of this strategic objective or, in particular, studying the evolution of the software industry into service-based organisations and identifying strategies, and technological roadmaps.* SELFMAN will target this area through its industrial partners who will study how the self-management principles developed in the project can be applied in industrial setting. They will evaluate the self-managing service architecture within an industrial standard platform (J2EE) and explore how a self-managing multi-tier application can be written in it. They will apply the self-management principles and algorithms to media-streaming. Finally, they expect to use the results of SELFMAN as input for their strategic decisions on future software development.

To summarize, SELFMAN is highly relevant to three major objectives of the Software and Services workprogramme, and is significantly relevant to the other two objectives as well.

4.1 Contributions with Respect to the State of the Art

SELFMAN will take a computer systems approach to self management. That is, we give a precise definition of self management in terms of computer system properties, namely configuration, fault tolerance, performance, and security. To make these properties self managing, we propose to design a system architecture and the protocols it needs. We consider that our approach is an effective one and that our project is a realistic way to achieve self management according

to our definition. But in the research community self management is sometimes defined in a broader way, to touch on various parts of artificial intelligence: learning systems, swarm intelligence (a.k.a. collective intelligence), biologically-inspired systems, and learning from the immune system [HERR05]. We consider that these artificial intelligence approaches are worth investigating in their own right. However, we consider that the computer systems approach taken by SELFMAN is a fundamental one that has to be solved, regardless of these approaches.

Now let us characterize the projected advances of SELFMAN with respect to the state of the art in computer systems. There are three areas to which we can compare SELFMAN:
1. *Structured overlay networks and peer-to-peer systems*. Current research on overlay networks focuses on algorithms for basic services such as communication and storage. The reorganizing abilities of structured overlay networks can be considered as *low-level* self management. SELFMAN will extend this to address high-level self management such as configuration, deployment, on-line updating, and evolution, which have been largely ignored so far in structured overlay network research.
2. *Component-based programming*. Current research on components focuses on architecture design issues and not on distributed programming. SELFMAN extends this to study component-based abstractions and architectural frameworks for large-scale distributed systems, by using overlay networks as an enabler.
3. *Autonomic systems*. Most autonomic systems projects study individual autonomic properties, specific self-managed systems, or focus on specific elements of autonomic behavior. Little research has considered the overall architectural implications of building self-managed distributed systems. The SELFMAN project is unique in this respect, combining as it does component-based system construction with overlay network technology into a service architecture for large-scale distributed system self management. The focus on both large-scale, loosely coupled systems, and architectural issues is a key differentiator of the SELFMAN project with respect to most of autonomic systems research.

We now present each of these areas in more detail and explain where the contribution of SELFMAN fits in.

The RAD Laboratory (Reliable, Adaptive, Distributed systems) was recently created at UC Berkeley and is funded by Google, Microsoft, and Sun. RAD intends to use research from statistical learning theory, control theory, and machine learning to improve the detection of problems in distributed systems, and to apply research from Recovery-Oriented Computing to provide fast recovery and reaction mechanisms that tie in to this. As such, RAD's vision overlaps with the SELFMAN vision. RAD proposes a specific solution path for improving reliability and adaptability for distributed systems. We consider that RAD and SELFMAN are complementary: RAD has a broad coverage of issues related to the construction of distributed applications, including considerations for Internet architecture and network monitoring. Also, RAD places a strong emphasis on statistical learning tools for assessing the behavior of distributed applications and integrating those in development tools and middleware. In contrast, SELFMAN has a more restricted scope, with little considerations for behavior assessment and diagnostics. SELFMAN will closely follow the RAD work and use its ideas where possible.

4.1.1 Structured Overlay Networks and Peer-to-Peer Systems

Research on peer-to-peer networks has evolved into research on structured overlay networks, in particular on Distributed Hash Tables (DHTs). The main differences between popular peer-to-peer systems and structured overlay networks are that the latter provide strong guarantees on routing and message delivery, and are implemented with more efficient algorithms [GHVR05]. The research on structured overlay networks has matured considerably in the last few years [ABER05, ELAN03, GHOD05, GHVR05, KAAS03, ROWS01, STOI01, ZHAO04]. Hardware infrastructures such as PlanetLab have enabled DHTs to be tested in realistically harsh environments [CHUN03]. This has led to structured peer-to-peer communication and storage infrastructures in which failures and system changes are handled gracefully.

Peer-to-peer systems such as Napster and Gnutella were initially built to enable using the resources of machines located at the edge of the network. For example, in Napster, music files were transferred directly between end users. Because of legal issues, completely decentralized systems, such as Gnutella, were built and popularized. These systems facilitated resource sharing in a completely decentralized way, i.e., there is no single point of failure. Much of the work up to this point was naïve and made use of techniques and algorithms known in computer science for decades, e.g., Gnutella uses a simple broadcast and converge-cast algorithm with limited horizon to search.

The technical inferiority of these systems started much interesting research on several frontiers. For instance, Adar and Huberman showed already in 2000 that approximately 60-70% of the users were abusing Gnutella, without sharing any resources. This opened up the field for research on using trust models efficiently in decentralized environments to *automatically* value a node based on its previous behaviour. Others focused on security in general, for instance [ABER04] showed how a system could protect itself from so called Sybil attacks, where one node joins the system many times with the purpose of gaining majority or complete control over parts of the system. We believe much of this research is useful in today's IT solutions where the virtual organization is dynamic and spans multiple sites, where security and trust will be of outmost importance. However, decentralized trust and security management has to be self managing, as the complexity for large-scale systems is too great for human administrators. Hence, self-management techniques should be an integral part of the infrastructure.

Another area of research on peer-to-peer systems that resulted in several self-managing properties is research on achieving *numeric*, *geographic*, and *administrative scalability*.

Research on numeric scalability was motivated by the deficiency of the Gnutella search method, which prevented many useful applications, as the search results were incomplete. Structured overlays, such as Chord, Pastry, P-Grid, Skipnet, Koorde, and Viceroy, showed that search can be done in a number of steps logarithmic to the system size, at the cost of relatively small routing tables. Some systems, such as Chord, DKS, and Pastry, need routing information logarithmic to the size of the system, while others, such as Viceroy, Koorde, and D2B, only need a constant size routing table. While the focus was on achieving numeric scalability, they all assumed the presence of *churn*, i.e. frequent joining, leaving, and failure of nodes. This fact, which we believe is underestimated, provides *self-management* of nodes in the system. Any peer-to-peer application, such as a file-server, will self-manage as servers are added or removed, without the need for any manual configuration. The system automatically re-organizes the data on the servers to adjust to the current load and capacity. This is what we call *low-level* self management.

Research on geographic scalability, under the name of *proximity/locality awareness*, has focused on making large-scale distributed peer-to-peer systems route efficiently, considering the physical limitations of the network. For example, a structured overlay should avoid routing from Europe to Japan and back to Europe to find a data item available on the LAN. This became especially apparent when [GUMM03] showed that in practice the majority of queries can be answered locally on the LAN. Today, most systems, such as Pastry, Tapestry, Kademlia, Land, and Tulip provide a *stretch* factor below 2, meaning that routing on the overlay is bounded by twice the "cost" of routing between the end hosts directly. Today, many distributed IT solutions, such as file systems or DNS, are manually and statically configured, and are not suited for dynamically evolving organizations. For example, in an AFS server, the placement of volumes is manually configured by system administrators. SELFMAN's configuration and load-balancing management can make a contribution in this area.

Research on administrative scalability has received less attention than many other research questions in peer-to-peer computing. SELFMAN can make a contribution in this area, to provide *content locality* and *middlebox traversal (NAT, firewalls)*. Content locality means that information about resources should be physically stored near the resources themselves. Consequently, information about resources can be found near the resources, minimizing the risk of a network partition preventing access to the resource. Traversing middleboxes entails that the overlay should be constructed without assuming that the network is fully connected, as firewalls and NATs might prevent connection being established between certain nodes. [MIS04] show how these problems can be attacked by connecting multiple heterogeneous overlays to form a hierarchy reflecting the organizational boundaries. SkipNet solves the problem by using order-preserving mappings in

the overlay, ensuring that nodes belonging to the same organization are neighbours on the overlay. More recently, [MONT05] and [SHAK05] show how the whole overlay can be self-adapted and re-reconstructed from scratch periodically, to form a topology that reflects the boundaries of the real world.

This brief summary of research on peer-to-peer overlays shows that they have indeed many *low-level* self-managing properties. We believe that these self-managing properties are necessary in any dynamic distributed application, and IT applications will greatly benefit from them. SELFMAN will extend this research by using a component model to make it possible to provide *high-level* self-managing properties. Note that because the system architecture is based on the component model, this means that the structured overlay network will itself be implemented in the component model.

4.1.2 Component-Based Programming

The main current de-facto standards in distributed software infrastructures, Sun's J2EE, Microsoft .Net, and OMG CORBA, provide a form of component-based distributed programming. Apart from the inclusion of publish-subscribe facilities (e.g. the JMS publish-subscribe services in J2EE), support for the construction of large-scale services is limited. Management functions are made available using the traditional manager agent framework [DMTF] but typically do not support online reconfiguration or autonomous behavior (which are left unspecified). Some implementations (e.g. JBoss [[FLEU03]) have adopted a component-based approach for the construction of the middleware itself, but they remain limited in their reconfiguration capabilities (coarse-grained, mostly deployment time, no support for un-planned software evolution).

Component models supported by standard platforms such as J2EE (the EJB model) or CORBA (the CCM model) -- see [SZYP02] for a recent survey -- are non-hierarchical (an assemblage of several components is not a component), and provide limited support for component introspection and dynamic adaptation. These limitations have been addressed in work on adaptive middleware (e.g. OpenORB [BLAI01], Dynamic TAO [KON00], Hadas [BENS01], that have demonstrated the benefits of a reflective component-based approach to the construction of adaptive middleware). In parallel, a large body of work on architecture description languages (e.g. ArchJava [ALDR02, ALDR03], C2 [MEDV99], Darwin [MAGE95], Wright [ALLE97], Rapide [LUCK95], Piccola [ACHE02], Acme [GARL00] or CommUnity [WERM01]) has shown the benefits of explicit software architecture for software maintenance and evolution. The component models proposed in these experimental prototypes, however, suffer from several limitations:
1. They do not allow the specification of component structures with sharing, a key feature required for the construction of software systems with resource multiplexing.
2. They remain limited in their adaptation capabilities, defining, for those that do provide such capabilities, a fixed meta-object protocol that disallows various optimizations and does not support different design tradeoffs (e.g. performance vs. flexibility).
3. Finally, and most importantly, they lack abstractions for building large distributed structures.

Compared to the current industrial and academic state of the art in component-based distributed system construction, the SELFMAN project intends to extend a reflective component-based model that subsumes the capabilities of the above models (it caters to points (1) and (2)) in order to address point (3).

4.1.3 Autonomic Systems

The main goal of autonomic system research is to automate the traditional functions associated with systems management, namely configuration management, fault management, performance management, security management and cost management [DMTF]. This goal is becoming of utmost importance because of increasing system complexity. It is this very realization that prompted major computer and software vendors to launch major R&D initiatives on this theme, notably, IBM's Autonomic Computing initiative and Microsoft's Dynamic Systems initiative.

The motivation for autonomic systems research is that networked environments today have reached a level of complexity and heterogeneity that make their control and management by human administrators more and more difficult. The complexity of individual elements (a single software element can literally have thousands of configuration parameters), combined with the brittleness inherent of today's distributed applications, makes it more and more difficult to entertain the presence of a human administrator in the "management loop". Consider for instance the following rough figures [GANE03]:

- One-third to one-half of a company's total IT budget is spent preventing or recovering from crashes.
- For every dollar used to purchase information storage, 9 dollars are spent to manage it.
- 40% of computer system outages are caused by human operator errors, not because they are poorly trained or do not have the right capabilities, but because of the complexities of today's computer systems.

IBM's autonomic computing initiative [IBM], for instance, was introduced in 2001 and presented as a "grand challenge" calling for a wide collaboration towards the development of computing systems that would have the following characteristics: self configuring, self healing, self tuning and self protecting, targeting the automation of the main management functional areas (self healing dealing with responses to failures, self protecting dealing with responses to attacks, self tuning dealing with continuous optimization of performance and operating costs). Since then, many R&D projects have been initiated to deal with autonomic computing aspects or support techniques. For example, we mention the following projects that are most relevant to SELFMAN:

- The Recovery-oriented Computing project [ROC] at UC Berkeley, which studies techniques for fault recovery through micro-reboot techniques.
- The OceanStore project [OCEANSTORE], also at UC Berkeley, which seeks to build a persistent, highly available and consistent data store able to scale to billions of users, is built upon a collection of untrusted servers.
- The Kinesthetics project [KINESTHETICS], at Columbia University, which seeks to develop a meta-architecture for building autonomic systems, including systems comprising legacy components.
- The Darwin project [DARWIN], at Imperial College, London, that develops a software architecture-based approach to the construction of self-monitoring and self-healing systems.
- The Smartfrog project [SMARTFROG], at HP Research Labs in Bristol, UK, that targets automatic deployment and configuration of distributed systems.
- The Swan project [SWAN], at INRIA, Alcatel, and France Telecom R&D, that develops novel models and algorithms for automated fault diagnosis and supervision.
- The Oceano project [OCEANO], at IBM J. Watson research center, which targets performance self-tuning for clusters, with dynamic resource allocation and the management of differentiated levels of services.
- The onCall project [NORR04], at Stanford University, which focuses on automatic performance and overload management in clusters.
- The BioOpera project [BAUS02], at ETH Zurich, which deals with fault management and automatic recovery of application workflows in cluster environments.

Compared to these projects, the uniqueness of SELFMAN is that it combines structured overlay networks with component models for the development of an integrated architecture for large-scale self-managing systems. Each complements the other: overlay networks support large-scale distribution, and component models support reconfiguration. None of the abovementioned projects provide such a combination, which gives a uniform architectural model for self-managing systems. Note also that many of the abovementioned projects are based on cluster architectures, whereas SELFMAN targets distributed systems that may be loosely coupled.

5. Potential Impact

We divide the potential impact of SELFMAN into technological impact and scientific impact. The technological impact mainly concerns the two industrial partners, France Telecom R&D and Stakk, and the Grid research community. The scientific impact concerns the foundational aspects of the project, the relationships with other European projects, and the dissemination activities.

5.1 Technological Impact

One of the key obstacles to deploying large-scale applications running on networks such as the Internet or company intranets is how to keep the application running despite changes in its computing environment, i.e., application management as defined in SELFMAN. Currently many specialized personnel are needed to keep large Internet applications running. SELFMAN will remove this obstacle, and thus enable the development of many more Internet applications and Internet-based companies that depend on such applications. In particular, France Telecom and Stakk are both interested in this aspect of SELFMAN.

France Telecom R&D expects the results of SELFMAN to support the future decisions of France Telecom on large-scale network architectures and service and IT platforms. France Telecom is currently confronted with the increasing software complexity and diversity of these architectures and platforms. The administration and exploitation of such systems is relatively labour-intensive due to round-the-clock monitoring and trouble-shooting in order to guarantee continuous high availability of the system as a whole. If the system itself becomes more autonomous and is able to take charge of its own monitoring and (re)configuration, this will represent important gains in operating costs for telcos (and ISPs). Of course, one might say that peer-to-peer VoIP systems such as Skype already have reduced costs because there is no centralized administration by specialist teams. This hides the fact that end-users of such systems take on some of the burden of administration of their individual system and also that guaranteeing end-to-end connectivity and availability in such peer-to-peer systems is still an open problem.

Stakk works in the area of P2P video streaming. Stakk is built on advances in P2P networks as well as optimization techniques. The main model adopted by Stakk is that P2P techniques are used to gather global information about peers watching live streams, e.g., bandwidth, delays, physical location, peer capacity, NATs, etc. A server farm is used to optimize the network connectivity between peers based on the gathered global information. Stakk sees in SELFMAN an opportunity to architect its server farm in a scalable and self-managed way. The traditional approaches for building such a server farm are inadequate for the desired functionality since there is more than one server farm in different network location, i.e. Point of Presence (POP). SELFMAN results would be beneficial to Stakk for architecting the scalability and management inside each POP as well as for inter-POP connectivity. Stakk will provide as input to the project information such as expected load scenarios and user collective user behavior and participate in the evaluation of he novel architectures developed in SELFMAN.

The main connections between SELFMAN and Grid research are through the CoreGrid 6FP Network of Excellence (of which UCL, KTH, INRIA, and ZIB are partners) and through the special expertise of ZIB, which is a major player in the Grid community. We expect to influence Grid research through these two channels. ZIB participated in the FP5 projects DataGrid, FlowGrid, and GridLab, and is participating in the 6FP SSA GridCoord and in numerous expert groups and workshops related to the Grid. In the DataGrid project, ZIB designed and implemented some primitive self-managing features, like the injection of so-called maintenance jobs. In the FlowGrid project, ZIB realized a complex Grid environment with distributed clusters, remote monitoring and control for the execution of computational fluid dynamics jobs for industrial use. Within GridLab, ZIB designed and implemented part of the Grid Application Toolkit, namely all

functions dealing with data handling (moving, copying, extraction, etc). All of these tasks are directly related to the goals of SELFMAN.

5.2 Scientific Impact

SELFMAN will build on the results of the PEPITO and SARDES projects. PEPITO is a 5FP project that involved several SELFMAN partners (UCL, KTH, INRIA) [GHOD05, GHVR05, ABER05]. PEPITO started with ideas from the peer-to-peer community and developed a mature technology for structured overlay networks. The structured overlay networks provide basic routing, communication, and storage services on top of a self-managing framework. This framework is scalable (any number of nodes) and handles node failures, node removals, and node additions automatically. SARDES is an INRIA project that is developing an advanced component model, the Fractal model, in collaboration with France Telecom R&D [BRUN04]. Both are SELFMAN partners. The Fractal model has advanced reflection and reification abilities. This allows applications to do self configuration. SELFMAN combines the technologies for structured overlay networks developed in PEPITO and for component models developed in SARDES, with the goal of doing self management for large-scale distributed systems. The service architecture built by SELFMAN on this combination should constitute one of the first comprehensive architectures for large distributed autonomic systems.

Several SELFMAN partners (UCL, KTH, France Telecom) are partners in the EVERGROW 6FP Integrated Project, which will run from 2004 to 2007. EVERGROW is exploring large scale-free networks, with two main foci. First, traffic analysis and simulation using its high-performance computing infrastructure (8 clusters). Two SELFMAN partners have EVERGROW clusters. Second, its collaboration with physicists for the analysis of large networks as complex systems. SELFMAN will exploit its connections to EVERGROW in both these areas.

SELFMAN will exploit the expertise of ZIB in data management for building the storage service of WP3. The ZIB Data Management System (ZIBDMS) is a strategic research project of the Computer Science Research group at ZIB. The project was started two years ago with the goal to provide a truly scalable, reliable and user-friendly access to distributed data in the Grid. ZIBDMS neither builds on databases nor on proprietary data formats. Rather, the basic unit of data is a file – hence it can be deployed in all kinds of environments (Unix, Windows). To provide user-friendly access to data, ZIBDMS has a means to handle attribute/value pairs. Hence, data is not retrieved via a hierarchical name space where files are identified through their location in a directory tree, but by means of specifying attribute/value pairs. Any amount and any type of attributes may be associated to a file, which makes it easy to adapt ZIBDMS to the various existing file systems. For true scalability in distributed environments, ZIBDMS builds on peer-to-peer architecture rather than existing environments (e.g. Globus, Unicore) with their inherent bottlenecks.

SELFMAN will demonstrate the benefits of its service architecture for self management by applying it to J2EE systems and by realizing one realistic two-tier application. For this application, SELFMAN will build on the results of the GORDA project, a recent IST project that studies replication protocols for database management systems, to cater for the needs of the database tier. One of the SELFMAN partners (INRIA) is a member of the GORDA project.

For its research implementation, SELFMAN will use the Mozart Programming System, which is a good representative of a disruptive technology. It is an example of advanced language design and implementation in which Europe has a leadership position [MOZART, MOZ04]. Two SELFMAN partners (KTH and UCL) are developers of Mozart. The Mozart system has a world-wide influence. It is downloaded several thousand times per year. It is featured in a comprehensive textbook and reference work published by MIT Press in 2004, which is already used for teaching in many universities worldwide [CTM04]. Mozart can be compared with other advanced programming systems whose development is primarily European, such as Erlang, Haskell, and Curry [ERLANG, HASKELL, CURRY]. Compared to these other systems, Mozart has a much broader support for programming concepts. Mozart has strong support for concurrent programming, for fault-tolerant distributed programming, and for constraint programming. In this regard, Mozart can be compared only to Erlang, which also has strong support for concurrent programming and fault-tolerant distributed programming. In our experience as programming language researchers, it is clear that an important direction

for the future is component-based programming using concurrent components. By extending Mozart with an advanced component model, SELFMAN will take a decisive step in this direction and advance the state of the art of language research in Europe. Work on the research implementation will build also on results obtained as part of the IST MIKADO project that provide a formal foundation for the Fractal model, in the form of new process calculi with localities. MIKADO is a 5FP IST project that involved two of the SELFMAN partners (INRIA, FT R&D). SELFMAN will build on these results for its work on component models.

5.3 Contributions to Standards

We expect that SELFMAN will influence standards in two areas:
1. First, SELFMAN will investigate to what degree self managing properties can be added to large-scale distributed systems built with J2EE. One approach we will use is to implement self-managing J2EE libraries with the same API as standard Java libraries. For example, a DHT library can be built with the Java Map interface. This will maximize the influence of our work on the Java community. The Java community can also benefit, through SELFMAN's contributions to the ObjectWeb consortium, from the extended Fractal model and ADL, and from the Java-based implementation of the SELFMAN service architecture. For instance, the results of the project on self-configuring components can influence standardization on Java modules and J2EE deployment.
2. Second, SELFMAN will influence Grid research through CoreGrid (partners UCL, KTH, INRIA, and ZIB) and through the special role of ZIB in the Grid community, as explained in Section 5. We expect that the addition of self management to the Grid standard will be strongly influenced by the results of SELFMAN.

6. Project Management and Exploitation/Dissemination Plans

6.1 Project Management

6.1.1 Top-level Management Structure

The project will be managed globally by a coordinator:
- The coordinator is Prof. Peter Van Roy from UCL. He is responsible for the scientific progress of the project, the coordination between workpackages, and all conflict resolution in the project.
- The coordinator will be assisted by Stéphanie Landrain from UCL. She will help the coordinator with all contractual obligations with respect to the Commission. This includes timely submission of contract information, cost statements and timetables, deliverables, progress and management reports, financial arrangements between the partners and the Commission, and information dissemination (Web sites, documents, etc.).

The coordinator may delegate specific tasks to other persons and inform the Commission of these delegations. In particular, the administrative work will be assisted by a part-time secretary paid for by the UCL management budget.

In cases where there is a clear conflict, its resolution will be done by a Project Board that consists of the coordinator, his administrative assistant, and one representative per partner (typically the partner manager). This representative will be chosen by each partner who will then distribute the name of this representative to all partners. After discussion of the problem and different possible solutions, the decision will be taken by a simple majority vote in the Project Board. In case of a tie, the coordinator has the casting vote. This decision will then be followed by all partners. The discussions and possible solutions will be documented in written form, as well as the final decision. This document will be distributed to all partners.

6.1.2 Internal Management Structure

Each workpackage has a lead contractor, which is the partner responsible for the correct operation of the workpackage. The lead contractor reports to the administrative and scientific coordinators. The lead contractors are as follows:
- WP1: KTH
- WP2: UCL
- WP3: ZIB
- WP4: INRIA
- WP5: FT R&D
- WP6: UCL

Each partner has one person, the partner manager, who is responsible for the correct operation of the partner with respect to all its tasks including that of lead contractor. The partner manager reports to the administrative and scientific coordinators. The partner managers are as follows (E-Plus has no partner manager):
- UCL: Peter Van Roy
- KTH: Seif Haridi
- INRIA: Jean-Bernard Stefani
- France Telecom R&D: Thierry Coupaye
- ZIB: Alexander Reinefeld
- Stakk: Sameh El-Ansary
- NUS: Roland Yap

6.1.3 Mechanisms for Assessment and Evaluation

The Project Board communicates regularly (by email, telephone, or meetings if necessary) to assess the progress of the project and to discuss corrective measures if the progress is insufficient. Objective evaluation of the project will be done according to the criteria listed in Section 2, supplemented by several other criteria. We consider the following criteria:

1. Evaluation according to the effectiveness of the self management mechanisms. This evaluation will mainly be done in WP5. We will evaluate the four axes of self management, namely configuration/upgrading, fault tolerance, performance, and security. This evaluation will be done in three ways:
   a. The first evaluation will be theoretical. We will do formal reasoning and proofs for the algorithms used in the self-managing architecture.
   b. The second evaluation will be a qualitative evaluation of industrial usage scenarios. These scenarios will primarily come from the industrial partners France Telecom R&D and Stakk, although we expect that the industrial experience of the other partners will provide more scenarios.
   c. The third evaluation will be a quantitative evaluation of industrial trace data. This will use the traffic generator of France Telecom R&D and the Stakk trace data to make quantitative verifications of the ability of the system.
2. Scientific results. We will evaluate the project according to the number and quality of publications in international journals and conferences and how these publications are cited.
3. Software results. We will evaluate the project according to the Open Source software that is released. This includes answering the following questions. Is the software of good quality? Can third parties use it and do third parties in fact use it?
4. Industrial impact. We will evaluate the influence of the project results on the strategic decisions of France Telecom and Stakk. This includes answering the following questions. Are project partners part of this decision process? Did the decisions, as far as can be ascertained, use project results and insights?

6.1.4 Year-by-year Measurable Assessment and Evaluation Criteria

The following gives a summary of the criteria to be used to evaluate the project each year. The consortium will show running demonstration software at each reviewing period including Month 12. The project will release software each year (including the first) and will show the functionality of the software with a demonstration. In general, all results of the project can be demonstrated at each reviewing period.

Month 12: The project has generated the following results:
1. The project website and Wiki are operational and satisfactory.
2. The low-level self-management mechanisms are part of the structured overlay network.
3. The basic component model is designed.
4. The architectural framework for the application architecture is designed.
5. The user requirements for managing application servers are known.
6. The results are published in high-quality scientific venues.

Month 24: The project has generated the following results:
1. The high-level self-management mechanisms are part of the structured overlay network.
2. The security architecture is part of the structured overlay network.
3. The J2EE and Mozart implementations of the structured overlay network are released.
4. The component model and the architectural framework for applications are released.
5. The reports on the four self properties are released.
6. The replicated storage service is released.
7. A simple database query mechanism is part of the storage service.
8. The specification of the demonstrator application is known.
9. The results are published in high-quality scientific venues.

Month 36: The project has generated the following results:
1. The software for the four self properties is released.
2. The self-managing dynamic WWW server is released for J2EE and Mozart.
3. The guidelines for development of self-managing applications are known.
4. The qualitative and quantitative evaluation of the self-management properties is done.
5. The results are published in high-quality scientific venues.
6. Evidence exists of satisfactory dissemination, e.g., influence on industrial partners' strategic decisions.

6.2 Plan for Using and Disseminating Knowledge

SELFMAN will disseminate its results in three main directions.
1. First, the software developed by SELFMAN will be released under an Open Source license. The Mozart system is already released under an Open Source license and we have found that this helps to spread its use and increase collaboration. The developments around a Java-based Architecture Description Language (ADL) will be disseminated as part of the Fractal project of the ObjectWeb open source middleware consortium [OBJE05]. Likewise, the implementation of the SELFMAN service architecture for J2EE systems will be made available through a new project of the ObjectWeb consortium.
2. Second, SELFMAN will publish its scientific results in major international conferences and journals. Major international conferences that are relevant to SELFMAN include **P2P200X** (IEEE International Conference on Peer-to-Peer Computing), **IPDPS** (IEEE International Parallel and Distributed Processing Symposium), **PODC** (Principles of Distributed Computing), **ICDCS** (International Conference on Distributed Computer Systems), **Middleware** (ACM/IFIP/USENIX International Middleware Conference), **EuroPar** (European Conference on Parallel Computing), **CBSE** (Component-Based Software Engineering), **ICAC** (International Conference on Autonomic Computing), **SelfMan** (IFIP/IEEE International Workshop on Self-Managed Systems & Services), **SRDS** (Symposium on Reliable Distributed Systems), **DSN** (International Conference on Dependable Systems and Networks), **GPCE** (International Conference on Generative Programming and Component Engineering), **ICSE** (International Conference on Software Engineering), **OOPSLA** (SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications), **ECOOP** (European Conference on Object-Oriented Programming), **POPL** (Principles of Programming Languages), **CONCUR** (Concurrency Theory). Major journals relevant to SELFMAN include IEEE **Distributed Systems Online**, **Distributed Systems**, **ACM TOCS** (Transactions on Computer Systems), **ACM TOPLAS** (Transactions on Programming Languages and Systems), **IEEE TSE** (Transactions on Software Engineering), **SPE** (Software: Practice and Experience), **HOSC** (Higher-Order and Symbolic Computation).
3. Third, SELFMAN will organize annual workshops. We intend to tie these workshops to major European conferences such as the P2P200X series of conferences on peer-to-peer research, the EuroPar series of conferences, the Middleware series of conferences, and especially the ECOOP series. ECOOP is interesting because it is a major force in object-oriented programming in Europe and because it encourages satellite workshops.

Concertation clause. The project will actively participate in the activities organised at programme level relating to the IST area with the objective of providing input towards common activities and receiving feedback (e.g. from clusters), offering advice and guidance and receiving information relating to IST programme implementation, standards, policy and regulatory activities, national or international initiatives, etc. The project participants will also commit themselves to support the organisation of an annual conference by providing papers, participating in technical programme committee, chairing sessions, reporting, etc. The project participants will help in developing dissemination material that can be used for communication towards the general public. For instance by developing a video demonstrating the results of the project, by having articles about the project in local newspapers, featuring the benefits of the research carried out for the

community reading the newspaper, or contributing to the development of public relations and state-of-the-art brochures. The project should set up a project website. This work is part of Task 6.4.

In addition, the project will create a project Wiki to encourage collaboration with all interested third parties. The project will also collaborate in technical Weblogs, such as Lambda the Ultimate, to discuss technical issues and disseminate its results. The project will identify other IST projects with common interests and study possible collaborations. The project will identify other projects at an international level with overlapping interests, such as the RAD Lab recently created at UC Berkeley. This will increase the critical mass of the research results.

We will encourage partners to commercialize results developed in SELFMAN when this is feasible. This will normally happen through contact with Innovation Centers in the partners' geographic regions. For example, at UCL this role is played by the Sopartec (the knowledge transfer and the seed capital investment fund of UCL). In Stockholm, KTH is working closely with STING (Stockholm Innovation & Growth).

6.2.1 Open Source Software

As mentioned above, a privileged exploitation path for the SELFMAN project is through the dissemination of its more mature software results as open source software (i.e. software made available under an open source license, as per the definition of the Open Source Initiative, see http://www.opensource.org). There are three stages in Open Source development: (1) seed software, (2) community building, and (3) active community. The goal of SELFMAN is to achieve (1): to have software of sufficient quality and ability that it is a true seed for further development. A useful seed will attract a user community. We will then initiate (2), but this process will go on beyond the end of the project since the software seeds will be released during and at the end of the project. The risk of SELFMAN to achieve (1) is that the software should be of sufficient quality when it is released into the Open Source community. Two open source consortia are of particular relevance: the Object Web consortium and the Mozart consortium.

The ObjectWeb consortium, hosted by one of the project partners (INRIA), focuses on the development of open source middleware. One of its flagship products is the JOnAS J2EE application server. The self management capabilities developed by the SELFMAN project should directly benefit the JOnAS code base, extending it with functionality it currently lacks, and opening opportunities to deploy JOnAS in new application scenarios such as so-called edge computing J2EE where application server tiers are duplicated and distributed across a wide-area network (in contrast, current application server tiers are merely distributed across local area networks, typically in PC cluster configurations). INRIA has a project with a French national research agency that involves application servers and includes Bull, who are interested in improving the management abilities of JOnAS. There are other companies who will be actively courted, such as a local SME in Grenoble interested in software management issues. More generally, the managed overlay network technology can benefit the ObjectWeb code base as a whole, in particular for integration in higher level middleware such as middleware for Web service orchestration or enterprise application integration, which typically lacks both scalability and management capabilities.

The Mozart consortium focuses on the development of the open source Mozart programming system, a multi-paradigm distributed programming platform which will be used by the SELFMAN project as the basis for one of its implementations. Two partners of the SELFMAN project (KTH and UCL) are members of the board that manages the Mozart consortium. We expect the results of the SELFMAN project in the area of distributed, component-based programming models and abstractions (such as e.g. reflective components, distributed connectors and operators) to benefit directly the Mozart platform, providing enhancements to the Mozart language, to the Mozart library, and to the Mozart distributed implementation.

6.3 Raising Public Participation and Awareness

In SELFMAN we intend to push a variety of techniques for public participation and awareness. The project participants will help in developing dissemination material that can be used for communication towards the general public. We intend to develop a video demonstrating the results of the project, by having articles about the project in local newspapers, featuring the benefits of the research carried out for the community reading the newspaper, or contributing to the development of public relations and state-of-the-art brochures. We intend to target activities such as national science fairs, small industry forums, educational institutes, expositions in public libraries, and technological museums to present popularized versions of the work. We consider that self management has a natural public appeal, because it leads to systems that have a more "life-like" or "proactive" behavior. This ties in to the public thirst for new and innovative technologies.

6.4 Intellectual Property Rights

Intellectual property is owned by the partner or partners that have generated it. Partners are required to inform their workpackage's lead contractor of any intellectual property rights acquired or applied for resulting from work in SELFMAN. All intellectual property rights generated within the project can be used by the partners for the purposes of the project. Nevertheless, we will encourage all partners to develop Open Source software when possible. The issues related to new knowledge will be addressed in the Consortium Agreement.

7. Detailed Implementation Plan

7.1 Introduction – General Description and Milestones

Multi-tier applications are the mainstay of industrial applications. A typical example is a two-tier application, consisting of a client talking to a server, which itself interfaces with a database (see top of the figure). The business logic is executed at the server and the application data and metadata are stored on the database. But multi-tier applications are brittle: they break when exposed to stresses such as failures, heavy loading (the "slashdot effect"), network congestion, and changes in their computing environment. In practice, these applications require intensive care by human managers to provide acceptable levels of service. This becomes especially cumbersome for large-scale systems. For example, deploying a distributed file system across several organizations requires much manual configuration, as does adding another file server to the existing infrastructure. If a file server crashes, most file systems will stop functioning or fail to provide full service. Instead, the system should reconfigure itself to use another file server. This desirable behavior is an example of self management.
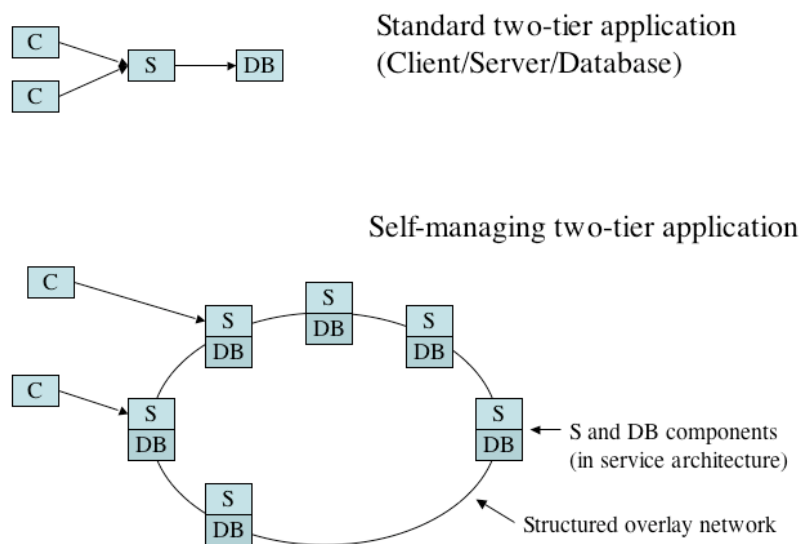
In the SELFMAN project, we will show how to make large-scale distributed applications self managing. We will implement a general architecture for these applications by combining research on structured overlay networks together with research on component models. These two areas each provide what the other lacks: structured overlay networks provide a robust communications infrastructure and low-level self-management properties, and component models provide the primitives needed to support dynamic configuration and enable high-level self-management properties. To show the effectiveness of this architecture, we will develop a J2EE demonstrator, where standard J2EE functions are deployed over a large scale network and are extended with self-management capabilities (see bottom part of the figure). We will evaluate the usefulness of the self-management abilities both quantitatively and qualitatively using industry data.

The foundation of SELFMAN will be a combination of a structured overlay network with a component model. Both areas have much matured in recent years, but they have been studied in isolation. It is a basic premise of SELFMAN that their combination will enable achieving self-management in large-scale distributed systems. This is first of all because

structured overlay networks already have many *low-level* self-management properties. Structured overlay network research has achieved efficient routing and communication algorithms, fault tolerance (automatically reconfiguring when a node fails), dynamic behavior (automatically reconfiguring when adding and removing nodes), proximity awareness, and distributed storage with replication (DHTs: Distributed Hash Tables). The reconfiguration and replication algorithms in fact are doing self management. However, almost no research has been done on deployment, upgrading, continuous operation, and other *high-level* self-management properties. By using components, we can add these high-level self-management properties. Recent research on component models, such as the Fractal model, is adding exactly those abilities that are needed for doing self management (such as reification and reflection abilities). But this research is still limited mostly to single machines or simple client/server applications.

The SELFMAN project will combine research on structured overlay networks and research on component models to achieve self management for large-scale distributed systems. The project partners have complementary expertise in these two areas. The project partners also have complementary expertise in data and storage management, which underlies a key service that we need for our core scenario, namely multi-tier applications. Note that since SELFMAN will investigate general principles of self management, it is not limited to two-tier or even multi-tier applications. Given their importance in industry, however, we consider that they are good targets for our work. With the business logic of the application and the application data and metadata, as before, our service architecture will run the application in a distributed setting over a structured overlay network. A main goal of the project is that very little changes are needed to the business logic's implementation or the data; we just have to change the infrastructure on which they run. In classical implementations of large multi-tier applications, the server and the database are implemented on dedicated servers or a cluster. This cluster architecture is tightly coupled, runs on one administrative domain, and is not scalable. On the other hand, SELFMAN proposes a loosely coupled architecture that runs on multiple administrative domains and is scalable. As a proof of concept, SELFMAN will implement a complete two-tier application based on a realistic storage/transaction service.

An important question for SELFMAN is on what platform do we build our implementation. The important issue is that the research results of SELFMAN must not be language or platform-specific. We require generic solutions, for example to fit both traditional object-oriented platforms (such as Java and C#) and recent disruptive platforms (such as Python, Ruby, and Mozart/Oz). Traditional platforms are designed for programming in the large and they encourage traditional software development processes. Disruptive platforms emphasize readability, small code size, and rapid development (e.g., extreme and agile programming). Both families have an important role for application development. In SELFMAN, we require that our results be scientifically well-founded and not be limited to a single family. In addition, we would like to compare the suitability of each family for build self-managing systems. We would like an answer to the question whethere there are inherent advantages or disadvantages to each family? A final reason for two implementation languages is that advanced component models already introduce a second language, an ADL (Architecture Description Language). We choose therefore to exploit this second language to our advantage.

We will choose one member of each family for the SELFMAN implementation. This choice has been done according to several criteria. One important criterium is that we would like to exploit the results of partner research in previous projects (such as the Fractal component model, developed by INRIA and France Telecom, the DKS middleware developed by KTH, and Mozart/P2PS system developed by UCL and others). This leads naturally to the following choice:

1. We choose Java and the J2EE platform to represent the traditional family. This will leverage all the work leading to DKS and Fractal. In addition, there already exist application servers that we can use as a base.
2. We choose Mozart as the representative of the disruptive family [MOZART]. We have complete control over the Mozart implementation including its virtual machine. That will allow us, for example, to implement migration for self tuning. An important property of Mozart and some other disruptive technologies is the support for lightweight thread concurrency. This tremendously simplifies the design of services as compared to the traditional event-driven model. A final point is the language itself: with Mozart we can extend the language to support self management directly and invisibly. With Java, this must be done with library calls and the addition of an architectural description language. With Mozart therefore we have the potential to do self management

without complicating the program and the development process, which is not possible with Java. That is, in Mozart the ADL has the potential to merge with the platform's language and "disappear".

It is interesting to compare the two approaches in terms of performance, readability, code size, and so forth. Task T6.5 will give the results of this comparison. In addition, we expect that these two implementations will provide synergy in other directions. Ideas will pass between the two implementations, thus strengthening both.

Workpackage Organization

SELFMAN is organized into six workpackages:

1. *Structured overlay network and basic mechanisms*. This workpackage will design and build a structured overlay network that provides basic self-management primitives (node failure/removal/addition) and that provides hooks for building self-managing applications on top of it. The structured overlay network will be built using the architecture of WP2.
2. *Service architecture and component model*. This workpackage will design and build an architecture for application development that has the ability to support self management. The architecture will be built using the structured overlay network of WP1 to provide the basic communication and self-management primitives.
3. *Self-managing storage and transactions*. This workpackage will design and build a storage service on top of the service architecture of WP2 and the structured overlay network of WP1. The storage service will do replication (implemented on the structured overlay network), provide a transactional interface, and provide a simple database interface. The storage service will support its own self management (node failure/removal/addition) using the primitives of WP4. The storage service will enable the applications of WP5.
4. *Self-management services*. This workpackage will implement the primitive services necessary for self-managing applications to be built using the architecture of WP2. These primitive services will be implemented using the detectors and actuators provides by the architecture of WP2. The primitive services will be used by the storage service of WP3 and the applications of WP5.
5. *Applications and evaluation*. This workpackage will implement a multi-tier application, in both a traditional format and a self-managing format. The application will be built using an advanced programming platform and an industrial platform. Comparisons and evaluation will be done between the traditional and self-managing formats, and between the advanced platform and the industrial platform.
6. *Project management*. This workpackage will ensure the smooth operation of the project. It will do progress assessment and evaluation, and conflict resolution.

We now discuss the important points of this organization. First of all, there is a close interaction between WP1 and WP2. It is one of the main premises of SELFMAN to exploit the synergy between structured overlay networks (WP1) and component models (WP2). Briefly, implementing a structured overlay network in a component model provides deployment, versioning, and upgrading facilities. In the other direction, the component model will incorporate the communication, routing, monitoring, and storage facilities of the structured overlay network as basic abstractions for the construction of large-scale self-managing systems.

A second point is the importance of WP3: the storage service. There are many possible services that could be built on top of the architecture of WP2. We have selected one service, a storage service with transactional interface, because of its importance for multi-tier applications. Given the limited size of SELFMAN, we consider that selecting one service and implementing it in depth is the most cost-effective way to use the project resources. There will be a close interaction between WP4, the self-management services, and WP3. The services of WP4 will be required by WP3, and the storage service from WP3 will be used to support the trading services developed in WP4

A third point is the importance of the implementation work and of WP5 on applications. The implementation work will be done in two platforms: an advanced research platform, Mozart, and a standard industrial platform, J2EE. The Mozart work will let us do more foundational work and explore the limits of our approach. The J2EE work will let us see in how far our work can be applied in an industrial setting. Having both in one project will let us do both foundational work and apply it in an industrial setting. For J2EE, it is important to see what could be done (in Mozart) in order to plan future developments. For Mozart, it is important to maintain a connection with an industrial platform, so that we can see how to migrate the work to that platform.

WP1: Structured Overlay Network and Basic Mechanisms

The main objective of this workpackage is to design and build a structured overlay network with two properties. First, it provides basic self-management primitives that are part of its own operation (node failure/removal/addition). Second, it provides the hooks for building self-managing applications on top of it (detectors and actuators). The structured overlay network will be built using the component model of WP2. Starting points for the work on this workpackage include the following implementations:

1. The DKS library, a structured overlay network built in J2EE [DKS05].
2. The P2PS library, a structured overlay network built in Mozart [PSPS05].

WP1 extends DKS and P2PS with the self-management hooks and by using the component model of WP2.

This workpackage is organized into five tasks:

1. Basic self management for structured overlay networks. This task will design the self-management mechanisms that are normally part of a structured overlay network. Note that these mechanisms will be implemented using the service architecture of WP2, in a factored way using component programming.
2. Self-management primitives in structured overlay networks. This task will design the detectors and actuators that are needed when building self-management mechanisms on top of structured overlay networks.
3. Security for structured overlay networks. This task has three parts: identifying threats, integrating security mechanisms into the overlay network, and building a monitoring system to detect security violations. Regarding security mechanisms, we need the ability to handle certificates at a fine grain (similar to capabilities) for the operations of the overlay network. We can also use the overlay network to store the revocation lists (which can become very large). Regarding monitoring, the LBOX system [WU05] is an example that provides high-level monitors which can interface in a secure fashion to the operating system kernel.
4. Structured overlay network in a standard component model. This task will implement the full structured overlay network, with both sets of primitives, in an industrial standard component model, namely J2EE.
5. Structured overlay network in an advanced component model. This task will implement the full structured overlay network in an advanced component model based on the Mozart system and designed in WP2.

WP2: Service Architecture and Component Model

The main objective of this workpackage is to design and implement the SELFMAN service architecture, a component-based architecture for large, self-managing distributed systems. The architecture relies on the structured overlay networks developed in WP1 for its basic distributed services. The component-based architecture shall comprise: (1) a reflective component-based computational model, (2) together with its formal semantics, and (3) a component-based architectural framework, comprising abstractions, design patterns, and basic infrastructure services.

The reflective component computational model will be defined as a programming language independent model for which two implementations will be studied as part of this workpackage: one based on Java and a complementary architecture description language (ADL), and one based on an extension of the multi-paradigm Mozart programming language. While the Java-based ADL approach is probably better for the dissemination and exploitation of the project software results in the medium term, the Mozart implementation will allow us to study more innovative programming language support and the combination of distributed component-based programming with other programming paradigms supported in Mozart.

Design patterns defined in the architectural framework will rely both on the structured overlay networks studied in WP1 and control and management patterns that begin to emerge as part of studies on autonomic systems architecture, greatly extending and generalizing the traditional Manager/Agent architecture at the heart of classical network and system management standards [DMTF]. Infrastructure services identified as part of the architecture will rely on basic structured overlay services studied in WP1.

Starting points for the work on this workpackage are well identified:

1. The Fractal component model [BRUN 04] designed and developed by INRIA and France Telecom, will provide an initial contribution to the SELFMAN computational model.
2. The Kell calculus [SCHM 04], developed by INRIA as part of the work on the IST MIKADO project, and the Mozart formal multi-paradigm computational model will serve as inputs to the definition of the formal semantics of the computational model.
3. The Jade framework, an architecture for self-managed cluster-size J2EE systems [BOUC 05] developed by INRIA, and the P2Pkit [P2PK05], a simple distributed component architecture built in Mozart using P2PS [P2PS05], will provide input to the development of the SELFMAN component-based architectural framework.

The workpackage is organized into three tasks, corresponding to the three elements of the SELFMAN service architecture:

1. Computation model. This task is responsible for defining the SELFMAN component-based computational model, and for developing its associated linguistic support, in Java and in Mozart. The key features we expect from the computation model include: notions of components and component connections as first-class elements, the ability to express component sharing (essential for expressing shared libraries and resources), reflective features allowing introspection and intercession at run-time, the ability to define and constraint component behavior (for self-management policies), and abstractions for handling partial failures including recovery actions.
2. Architectural framework. This task is responsible for specifying the architectural framework that embodies the SELFMAN service architecture, and for developing associated supporting tools such as e.g. specific ADL modules. We expect to include architectural patterns that cover at least the following: elements for the construction of distributed feedback and feedforward control loops (which are at the heart of management behavior) and elements for the construction of distributed self-healing services (including possible failures in the control loops themselves). We also expect some tools, including tools for automatically generating helper functions and components, e.g., for automatic deployment, system instrumentation, and dynamic reconfiguration. We expect to complement the linguistic support of the first task with more specialized languages and programming abstractions.
3. Formal semantics. This task is responsible for defining the formal semantics of the SELFMAN computational model. In our experience, it is essential to have a simple formal semantics when developing new abstractions (such as a computation model and architectural framework). This guarantees that there is no unexpected bad behavior.

Two of the important questions that this workpackage will provide answers to are concurrency and communication. How will concurrency be managed between components: when should components be sequential and when should they be concurrent? How does this fit with the concurrency abilities of the underlying implementation platform? I.e., J2EE is built using Java and supports only coarse-grain concurrency (few threads). Mozart supports fine-grain concurrency (thousands of threads). For communication, we will determine to what degree the communication will be synchronous or asynchronous, and tightly coupled or loosely coupled. For example, we may use a tuple space abstraction, which allows components to communicate in a very loose fashion, where the source and destination components do not even know the identity of the other component.

WP3: Self-Managing Storage and Transactions

The main objective of this workpackage is to design and build a storage service on top of the service architecture of WP2 and the structured overlay network of WP1. The storage service will do replication (implemented over the structured overlay network), provide a transactional interface, and provide a simple database interface. The storage service will build on previous work by project partners:

1. The ZIBDMS database management system that is being developed by ZIB.
2. The experience in designing transaction protocols over structured overlay networks of KTH and UCL [MESA05].

The simple database interface is not intended to be a competitor to commercial databases (that would be impossible in a project of the size of SELFMAN) but to provide a proof-of-concept of a distributed database built on top of a structured overlay network. Such a distributed database does not yet exist and will be one of the contributions of SELFMAN. The storage service will support its own self management (node failure/removal/addition) using the primitives of WP2 and WP4. The storage service will enable the applications of WP5. Note that the storage service is an example of a realistic self-managing service built on top of the service architecture defined by WP1 and WP2. As such, the storage service will serve to validate this architecture even before we build applications with it. The storage service will also be used for the component trading service that will be built in WP4.

WP4: Self-Management Services

The main objective of this Workpackage is to develop support for the implementation of selected autonomic services, namely self-configuration, self-healing, and self-tuning services. We believe self-healing and self-tuning services require a good deal of knowledge of the system behavior. Self tuning in particular requires a knowledge of the system performance under various conditions, especially overload situations. For this reason, self healing and self tuning aspects will be studied in close relation with the application scenarios developed as part of WP5. Also, the project will focus only on certain key mechanisms pertaining to these two services, namely fault-tolerance and repair management mechanisms for self-healing, and load balancing and overload management mechanisms for self tuning.

The workpackage is organized in four tasks, corresponding to the services we have targeted:
1. Support for self configuration. This task is responsible for developing mechanisms and infrastructure support for the development of self-configurable systems, including support for on-line system upgrade at both middleware and application levels. Configuration, as understood in this task, covers both traditional distributed deployment and installation, as well as dynamic update and on-line reconfiguration. In the general case, configuration processes can embody quite complex distributed workflows, since they must take into account partial failures, distributed synchronization and consistency constraints, as well as various constraints such as versioning constraints, capacity constraints for component placement, component dependencies, etc. Because of this, support for self configuration should contain: a meta-model for component packages (packages as first-class entities), a distributed trading service (selection of component packages on a large scale), and linguistic support for deployment and reconfiguration workflows.
2. Support for self healing. This task is responsible for developing mechanisms and infrastructure support for the development of self-healing and self repairing systems, including support for multi-level fault tolerance and repair management. The task will comprise the following: fault detection and fault tolerance algorithms and tools, algorithms for configuration repair, replication algorithms, and policies for self repair. This task will study the construction of self-healing systems by means of replicated control loops for configuration repair.
3. Support for self tuning. This task is responsible for developing mechanisms and infrastructure support for the development of self-tuning and self-optimizing systems, including support for multi-level load balancing and overload management. The task will comprise the following: empirical performance models and profiling, algorithms for load balancing and load adaptation, and formal analysis of stability.
4. Support for self protection. We will investigate what kind of higher-level security mechanisms are appropriate for self management. This links up to what kinds of end-to-end security requirements are needed at the application level. Some of the basic security questions at the overlay network level are discussed in [SIT02] but this does not deal with higher level mechanisms. Clearly some kind of detection system is important. [SUF05] shows that many intrusion detection systems can suffer from subtle attacks against the intrusion system itself. So one line of investigation is whether a reliable detection mechanism can be built at the self management level. This would also need to integrate with a self monitoring infrastructure (as will be provided by WP1).

These services will be built using the detectors and actuators provided by the architecture of WP2. The primitive services will be used by the storage service of WP3 and the applications of WP5. This workpackage will involve many partners to build the primitive self-managing services, each partner contributing its expertise in collaboration with INRIA, which has the most expertise in self management. This workpackage will build on the previous experience of INRIA and France

Telecom R&D on building self-managing services on top of a component model in the SARDES project. This workpackage goes beyond SARDES in that it is based on a structured overlay network, which is designed and implemented in WP1. This workpackage will also build on the experience of KTH in formally analyzing structured overlay networks using techniques from theoretical physics, in the EVERGROW project [AURE04]. We expect to understand the interactions between self-management services and how to ensure stability in application execution.

WP5: Application Requirements and Evaluations

This workpackage has two important roles. First, it will study the requirements for management of application-hosting environments, also called application servers, with the goal of scaling it up to large numbers of machines that may be loosely coupled. Second, it will implement a multi-tier application, in both a traditional format and a self-managing format. The application will be built using an advanced programming platform and an industrial platform. We propose a dynamic serverless WWW server, which is a two-tier application in which software components can be changed at run-time, as the application. One of the project partners, KTH, already has experience implementing a static WWW server on top of the DKS platform. This static server has no self-management mechanisms beyond those of the structured overlay network DKS itself. In this workpackage we will compare the traditional and self-managing design for the WWW server. We will also compare the advanced Mozart platform and the industrial J2EE platform. Finally, we will use the traffic generator of France Telecom R&D and the trace data of Stakk to measure the effectiveness of the application and also of the service architecture.

There will be continuous feedback between WP5 and the two workpackages WP3 and WP4. The needs of WP5 that are felt directly during the demonstrator application will affect both the storage service (WP3) and the high-level self management mechanisms (WP4). Although we will do our best to make the results of WP3 and WP4 be the needed ones, practice shows that feedback between applications and service development is essential for getting best results.

As an addition in the project's third year, we propose to build a distributed theater demonstrator application for mobile devices. It has become possible to add this application because of the successful deployment of the self-managing transactional storage in the second year. The application supports two kinds of users: actors and spectators. Spectators mostly receive information while actors both send and receive information. Actors interact through a simple shared virtual space that resembles a user interface canvas. The number of actors remains relatively small whereas the number of spectators can become extremely large. To avoid the accessibility and load problems of client/server architectures, we propose to make the application serverless and based on a structured peer-to-peer network. The application will use the Mozart implementation of the scalable storage service of WP3. We will port Mozart to gPhone mobile devices and use Wifi and Bluetooth for the lower-level communication layer. The application requires up to 10 gPhone mobile devices (HTC Dream G1) taken from the UCL budget for the proof-of-concept. The gPhone software development kit is free of charge.

This application is an opportunity to test and improve the dynamic connectivity and robust communication of our P2P storage subsystem while it maintains coherence of a shared updated virtual space. We will use three tests at different scales: small (simple communication), medium (small P2P network with less than 10 devices), and large (10 or more devices as actors together with many external networked computers as spectators).

7.2 Planning and Timetable

```
                                                              0        12       24       36
```

WP1: Structured overlay network and basic mechanisms
T1.1 Low-level self-management
T1.2 High-level self-management primitives
T1.3 Security for structured overlay networks
T1.4 Structured overlay network in a standard component model
T1.5 Structured overlay network in an advanced component model

WP2: Service architecture and component model
T2.1 Component-based computation model
T2.2 Self-management architectural framework
T2.3 Formal semantics of computation model

WP3: Self-managing storage and transactions
T3.1 Formal models for transactions
T3.2 Replicated storage service
T3.3 Simple database query layer

WP4: Self-management services
T4.1 Support for self configuration
T4.2 Support for self healing
T4.3 Support for self tuning
T4.4 Support for self protection

WP5: Application requirements and evaluations
T5.1 User requirements for application servers
T5.2 Self-managing dynamic WWW server on industrial platform
T5.3 Self-managing dynamic WWW server on research platform
T5.4 Testing and evaluation using industrial trace data
T5.5 Traffic generation for WWW server evaluation
T5.6 Evaluation of security mechanisms
T5.7 Guidelines for third-party developers
T5.8 Self-managing distributed theater app on mobile devices

WP6: Management, dissemination, and exploitation
T6.1 Project management
T6.2 Workpackage management
T6.3 Dissemination and exploitation
T6.4 Synergies and collaborations
T6.5 Assessment, evaluation, and lessons learned

7.3 Graphical Presentation of Workpackages

→ Is input to

```
                                    ┌─────────────────────┐
                                    │        WP1          │
                                    │ Structured overlay  │
┌──────────────────────┐           │  network and basic  │
│        WP6           │           │    mechanisms       │
│    Management,       │           └─────────────────────┘
│ dissemination, and   │                    ↕
│   exploitation       │           ┌─────────────────────┐
└──────────────────────┘           │        WP2          │
                                    │ Service architecture│
                                    │  and component      │
                                    │     model           │
                                    └─────────────────────┘

┌──────────────────────┐           ┌─────────────────────┐
│        WP3           │  ──────→  │        WP4          │
│   Self-managing      │           │  Self-management    │
│    storage and       │  ←──────  │    services         │
│    transactions      │           └─────────────────────┘
└──────────────────────┘

                    ┌─────────────────────┐
                    │        WP5          │
                    │    Application      │
                    │ requirements and    │
                    │    evaluations      │
                    └─────────────────────┘
```
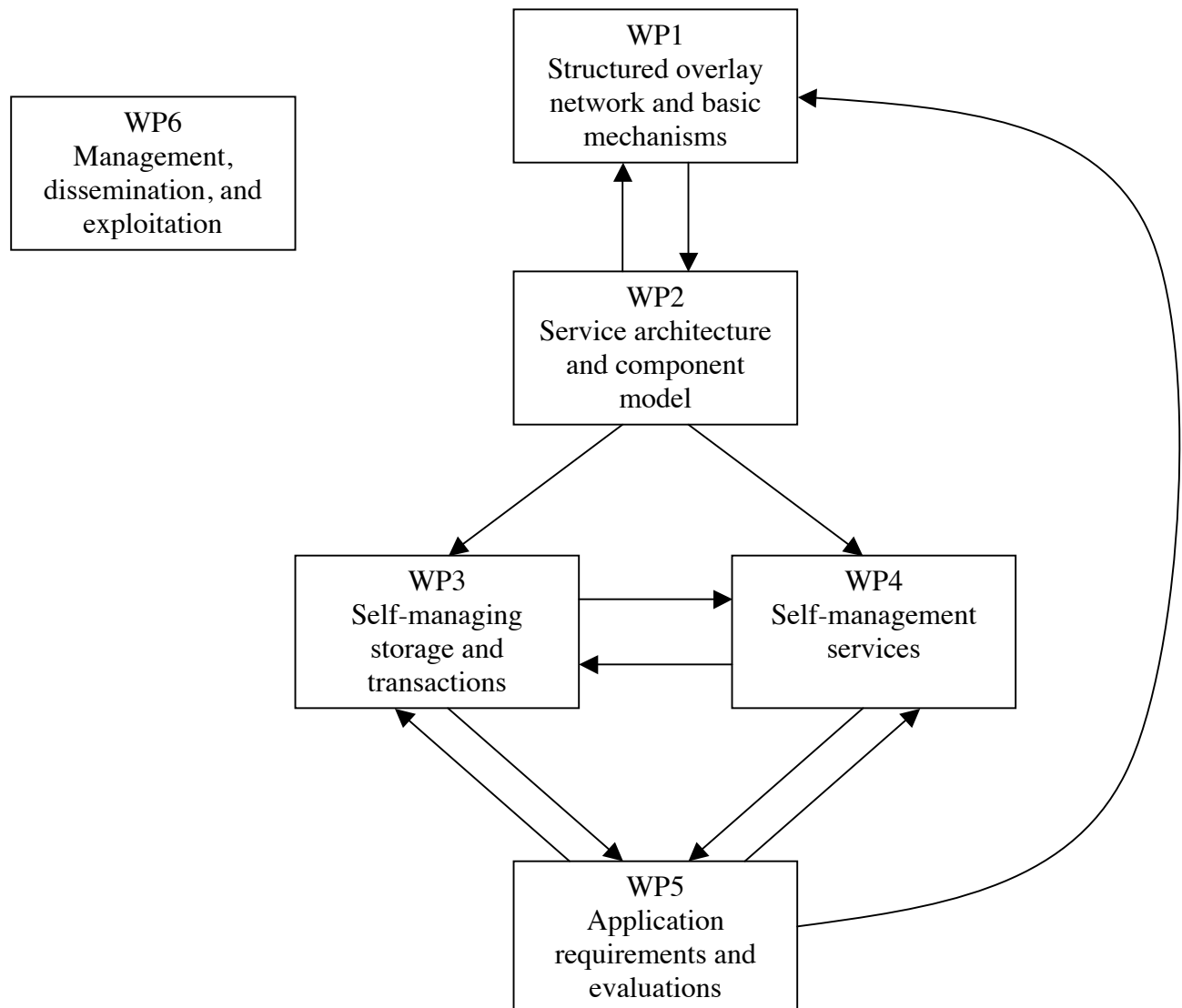
7.4 Workpackage List

Work-package No	Workpackage title	Lead contractor No	Person-months	Start month	End month	Deliverable No
WP1	Structured overlay network and basic mechanisms	2	51	0	24	D1.1, D1.2, D1.3a, D1.3b, D1.4, D1.5
WP2	Service architecture and component model	1	70	0	24	D2.1a, D2.1b, D2.1c, D2.2a, D2.2b, D2.2c, D2.3a, D2.3b
WP3	Self-managing storage and transactions	5	43	0	36	D3.1a, D3.1b, D3.1c, D3.2a, D3.2b, D3.3a, D3.3b
WP4	Self-management services	3	76	12	36	D4.1a, D4.1b, D4.1c, D4.2a, D4.2b, D4.2c, D4.3a, D4.3b, D4.3c, D4.4a, D4.4b
WP5	Application requirements and evaluations	4	47	0	36	D5.1, D5.2a, D5.2b, D5.3, D5.4a, D5.4b, D5.6, D5.7
WP6	Management, dissemination, and exploitation	1	30	0	36	D6.1a, D6.1b, D6.1c, D6.1d, D6.3, D6.5a, D6.5b, D6.5c
	TOTAL		317			

7.5 Deliverables List

Note: for WP6 the effort (pm) for the management part (5 pm) does not add up to the total man-power (16 pm) because most of the management work does not result in numbered deliverables (e.g., the administrative work). Deliverables D6.1a-d are management deliverables. The deliverables D6.1b-d are concerned with workshops; they will consist of the workshops themselves and a report on the workshop and its conclusions either in paper or electronic format (e.g. CD).

Deliv. No	Deliverable title	Effort (PM)	Deliv. date	Natu re	Dissem. level
D1.1	Report on low-level self-management primitives for structured overlay networks	12	12	R	PU
D1.2	Report on high-level self-management primitives for structured overlay networks	16	24	R	PU
D1.3a	First report on security for structured overlay networks	6	12	R	PU
D1.3b	Final report on security for structured overlay networks	5	24	R	PU
D1.4	J2EE library of SELFMAN structured overlay network	4	24	P	PU
D1.5	Mozart library of SELFMAN structured overlay network	8	24	P	PU
D2.1a	Report on basic computation model	6	12	R	PU
D2.1b	Report on computation model with self-management primitives	6	24	R	PU
D2.1c	Component-based computation model	12	24	P	PU
D2.2a	Report on architectural framework specification	9	12	R	PU
D2.2b	Report on architectural framework tool support	9	24	R	PU
D2.2c	Architectural framework	19	24	P	PU
D2.3a	Report on formal operational semantics (component and reflection)	4	12	R	PU

Deliv. No	Deliverable title	Effort (PM)	Deliv. date	Nature	Dissem. level
D2.3b	Report on formal operational semantics (distributed abstractions)	5	24	R	PU
D3.1a	First report on formal models for transactions over structured overlay networks	3	12	R	PU
D3.1b	Second report on formal models for transactions over structured overlay networks	3	24	R	PU
D3.1c	Final report on formal models for transactions over structured overlay networks	2	30	R	PU
D3.2a	Replicated storage service over a structured overlay network	18	24	P	PU
D3.2b	Report on replicated storage service over a structured overlay network	6	36	R	PU
D3.3a	Simple database query layer for replicated storage service	8	24	P	PU
D3.3b	Report on simple database query layer for replicated storage service	3	36	R	PU
D4.1a	First report on self-configuration support	4	24	R	PU
D4.1b	Second report on self-configuration support	4	36	R	PU
D4.1c	Self-configuration support	12	36	P	PU
D4.2a	First report on self-healing support	3	24	R	PU
D4.2b	Second report on self-healing support	3	36	R	PU
D4.2c	Self-healing support	8	36	P	PU
D4.3a	First report on self-tuning support	5	24	R	PU
D4.3b	Second report on self-tuning support	5	36	R	PU
D4.3c	Self-tuning support	11	36	P	PU
D4.4a	First report on self-protection support	10	24	R	PU

Deliv. No	Deliverable title	Effort (PM)	Deliv. date	Natu re	Dissem. level
D4.4b	Self-protection support	11	36	P	PU
D5.1	Report on user requirements for application servers	3	12	R	PU
D5.2a	Design specification of self-managing dynamic WWW application	4	24	R	PU
D5.2b	Self-managing dynamic WWW server application for J2EE	6	36	P	PU
D5.3	Self-managing dynamic WWW server application for Mozart	5	36	P	PU
D5.4a	Qualitative evaluation of self-management properties	7	36	R	PU
D5.4b	Quantitative evaluation of self-management properties	8	36	R	PU
D5.6	Evaluation of security mechanisms	3	36	R	PU
D5.7	Guidelines for developing self-managing applications	8	36	R	PU
D5.8	Self-managing distributed theatre app on mobile devices	3	36	P	PU
D6.1a	Project website and Wiki. (After M6, continuously updated)	2	1, 6, ...	P	PU
D6.1b	First project workshop	1	12	O	PU
D6.1c	Second project workshop	1	24	O	PU
D6.1d	Third project workshop	1	36	O	PU
D6.3	Dissemination and use report	4	36	R	PU
D6.5a	First progress and assessment report with lessons learned	3	12	R	PU
D6.5b	Second progress and assessment report with lessons learned	3	24	R	PU
D6.5c	Final progress and assessment report with lessons learned	4	36	R	PU

7.6 Workpackage Descriptions

Workpackage 1 Description

Workpackage number	WP1	Start date or starting event:				0	
Workpackage title: Structured overlay network and basic mechanisms							
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	14	10	8	0	0	0	17	2

Objectives

To design and implement a structured overlay network that provides the basic self management abilities of node failure/removal/addition, and that provides the self management primitives (detectors and actuators) needed by the service architecture of WP2.

Description of work

T1.1 Low-level self-management for structured overlay networks (UCL:4, KTH:2, INRIA:2 NUS:4) This task will design the self-management mechanisms and their algorithms that are normally part of a structured overlay network. These mechanisms will handle node failure, node removal, node addition, state monitoring, and a simple threat model.

T1.2 High-level self-management primitives in structured overlay networks (UCL:4, KTH:2, INRIA:6, NUS:4) This task will design the required primitives for configuration, versioning, updating. These primitives are detectors and actuators; they are needed for systems built *on top* of the structured overlay network. These will be used by the service architecture of WP2 when interacting with the structured overlay network, by the storage service of WP3, and by the self management services of WP4.

T1.3 Security for structured overlay networks (NUS:9, KTH:2) This task will look at (1) identifying the relevant threats, (2) integrating security mechanisms into the overlay network, and (3) building a distributed monitoring system for detection of security violations.

T1.4 Structured overlay network in a standard component model (KTH:2, Stakk:2) This task will design and implement the above structured overlay network in an industrial standard component model, namely J2EE.

T1.5 Structured overlay network in an advanced component model (UCL:6, KTH:2) This task will design and implement the above structured overlay network in an advanced component model based on the Mozart system.

Deliverables

D1.1 Report on low-level self-management primitives for structured overlay networks. This deliverable reports on the first 12 months of T1.1 and T1.2. **(M12)**

D1.2 Report on high-level self-management primitives for structured overlay networks. This deliverable reports on the second 12 months of T1.1 and T1.2. **(M24)**

D1.3a First report on security in structured overlay networks. **(M12)**

D1.3b Final report on security in structured overlay networks. **(M24)**

D1.4 J2EE library of SELFMAN structured overlay network. **(M24)**

D1.5 Mozart library of SELFMAN structured overlay network. **(M24)**

Milestones and expected result

M12: Understand how to do structured overlay network with component model.

M24: Finished structured overlay network with component model.

Workpackage 2 Description

Workpackage number	WP2		Start date or starting event:			0		
Workpackage title: Service architecture and component model								
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	17	12	19	11	11	0	0	0

Objectives

To design and implement a distributed component architecture with the basic primitives needed for self management. The component architecture uses the structured overlay network of WP1 as its foundation.

Description of work

T2.1 Component-based computation model (UCL:6, INRIA:7, KTH:6, FT R&D:5) This task will define the computation model used in the project and will develop linguistic support for it in the form of architecture description and programming languages and tools. The basic assumption is that the computation model needs to be component-based to allow the construction of self-configurable systems, with components being units of deployment, configuration, and fault confinement. One of the starting points for the task will be the Fractal component model [BRUN04] developed by two of the project partners (INRIA and FT R&D). For linguistic support, the task will study the extension of the Mozart programming language and environment with first class components, and will define in parallel an architecture description language and a Java mapping for the computation model.

T2.2 Self-management architectural framework (UCL:6, INRIA:8, KTH:6, ZIB:11, FT R&D:6) This task will develop the SELFMAN self-management architecture in the form of a software architectural framework together with associated tools. The core of the framework should consist in a set of architectural patterns and identified supporting services from the work in WP1.

T2.3 Formal semantics of computation model (INRIA:4, UCL:5) This task will develop a formal operational semantics for the SELFMAN computation model. A formal operational semantics is necessary to ensure a consistent definition of the model and to allow reasoning about the properties of a system built according to the model. Starting points for the task will include the Kell calculus [SCHM04] developed by one of the partners (INRIA) as part of the IST project Mikado, and ongoing work at one of the partners (UCL) for the introduction of a notion of membrane in Mozart. We expect the results from this task to provide an extension of the formal multi-paradigm computation model at the heart of the Mozart system.

Deliverables

D2.1a Report on basic computation model. **(M12)**

D2.1b Report on computation model with self-management primitives. **(M24)**

D2.1c Component-based computation model. **(M24)**

D2.2a Report on architectural framework specification. **(M12)**

D2.2b Report on architectural framework tool support. **(M24)**

D2.2c Architectural framework. **(M24)**

D2.3a Report on formal operational semantics (component and reflection). **(M12)**

D2.3b Report on formal operational semantics (distributed abstractions). **(M24)**

Milestones and expected result

M12: Understand architectural framework with component model using SON.

M24: Finished architectural framework with component model using SON.

Workpackage 3 Description

Workpackage number	WP3	Start date or starting event:			0		
Workpackage title: Self-managing storage and transactions							
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	6	11	0	8	15	0	0	3

Objectives

To design and build a self-managing storage service that provides data replication and the ability to perform transactions. This service will be built on top of the structured overlay network of WP1 and using the component model of WP2. This service is the foundation of the multi-tier application of WP5.

Description of work

T3.1 Formal models for transactions over a structured overlay network (ZIB:3, UCL:3, FT R&D:2)
This task will investigate how to do transactions over a structured overlay network. These will likely be different from classical transactions. One of the important research questions is to resolve the tension between the distributed system and the needs of the application. We will design different compromises and see which are appropriate for applications.

T3.2 Replicated storage service over a structured overlay network (ZIB:9, KTH:6, UCL:3, Stakk:2, FT R&D:4) This task will design a replicated storage service using the component model of WP2, running over the structured overlay network of WP1. This task will do both a J2EE and a Mozart implementation.

T3.3 Simple database query layer (KTH:5, ZIB:3, Stakk:1, FT R&D:2) This task will implement a simple database query layer on top of the replicated storage service. This will let us build the applications of WP5 at little effort. Note that full database functionality is not needed for the proof of concept.

Deliverables

D3.1a First report on formal models for transactions over structured overlay networks. **(M12)**

D3.1b Second report on formal models for transactions over structured overlay networks**. (M24)**

D3.1c Final report on formal models for transactions over structured overlay networks. **(M30)**

D3.2a Replicated storage service over a structured overlay network. **(M24)**

D3.2b Report on replicated storage service over a structured overlay network. **(M36)**

D3.3a Simple database query layer for replicated storage service. **(M24)**

D3.3b Report on simple database query layer for replicated storage service. **(M36)**

Milestones and expected result

M12: Understand how to do transactions over structured overlay network.

M24: Finished replicated storage service over structured overlay network.

M24: Finished query layer for replicated storage service.

Workpackage 4 Description

Workpackage number	WP4		Start date or starting event:			12		
Workpackage title: Self-management services								
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	6	14	19	3	15	0	17	2

Objectives

To design and implement the self-management services needed by multi-tier applications. This includes service configuration, reconfiguration, deployment, upgrading during execution, and so forth. We will formalize the self-management services and investigate under what conditions their behavior is convergent.

Description of work

T4.1 Support for self configuration (INRIA:5, UCL:6, FT R&D:3, KTH:5, Stakk:1) This task will develop support for the implementation of self-configuring distributed systems, in the form of distributed services and tools refining the generic self-management architectural framework developed in WP2. Configuration, as understood in this task, covers traditional distributed deployment and installation, as well as dynamic update and on-line reconfiguration.

T4.2 Support for self healing (INRIA:5, KTH:9) This task will develop support for the implementation of self-healing distributed systems, in the form of distributed services and tools refining the self-management architectural framework developed in WP2. The task will consider hardware and software faults.

T4.3 Support for self tuning (INRIA:5, ZIB:15, Stakk:1) This task will develop support for the implementation of self-tuning systems. Considering the vast scope of performance management, the task will be driven primarily by the needs of application scenarios developed in WP5, and will develop performance management functions and self-tuning features primarily dedicated to the handling of these scenarios.

T4.4 Support for self protection (INRIA:4, NUS:17) This task will develop support for simple self-protection mechanisms and policies, based on a simple threat model and trust model. This task is necessary to achieve realistic results in the project. The final results of this task will be reported in D5.6.

Deliverables

D4.1a First report on self-configuration support. **(M24)**

D4.1b Second report on self-configuration support. **(M36)**

D4.1c Self-configuration support. **(M36)**

D4.2a First report on self-healing support. **(M24)**

D4.2b Second report on self-healing support. **(M36)**

D4.2c Self-healing support. **(M36)**

D4.3a First report on self-tuning support. **(M24)**

D4.3b Second report on self-tuning support. **(M36)**

D4.3c Self-tuning support. **(M36)**

D4.4a First report on self-protection support. **(M24)**

D4.4b Self-protection support. **(M36)**

Milestones and expected result

M24: Understand how to incorporate self-* services on architectural framework.

M36: Finished self-* services on architectural framework.

Workpackage 5 Description

Workpackage number	WP5		Start date or starting event:			0	
Workpackage title: Application requirements and evaluations							
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	9	5	6	9	11	0	3	4

Objectives

To build a two-tier application using the service architecture of WP2 and the storage service of WP3, self-managed using the services of WP4. To evaluate and compare standard and self-managing versions of the application. To evaluate and compare the J2EE and Mozart implementations of the application.

Description of work

T5.1 User requirements for application servers (FT R&D:3 Stakk:1) In this task, the industrial partners will study the requirements for managing the hosting environments used to host applications. These requirements will be input for the self-management architecture of the project.

T5.2 Self-managing dynamic WWW server on industrial platform (ZIB:4, KTH:4) This task will implement a WWW server as a self-managing application using the extended J2EE environment. The task will then evaluate the usefulness of the self-management infrastructure. The application will be based on the same business logic and data as a non-self-managing application. Note that we may change the application if a more advantageous one appears.

T5.3 Self-managing dynamic WWW server on research platform (UCL:2, INRIA:5) This task will implement the WWW server on the Mozart platform. The task will then compare the J2EE and the Mozart implementations and make conclusions over future development of self management and the future evolution of each platform.

T5.4 Testing and evaluation using industrial trace data (Stakk:2, ZIB:3) This task will evaluate the efficiency of self management using trace data on P2P video streaming usage from Stakk. This task will start early so as to guide the development of workpackages WP3 and WP4.

T5.5 Traffic generation for WWW server evaluation (FT R&D:6, ZIB:1, KTH:1, UCL:1, INRIA:1) This task will provide a traffic generator and relevant probes, based on the CLIF load injection framework. This will be used to emulate the actual usage of the WWW server applications to observe their behavior. The results will be part of deliverables D5.4a and D5.4b.

T5.6 Evaluation of security mechanisms (NUS:3) This task will evaluate the security mechanisms of WP1 (overlay networks) and WP4 (higher level) in the context of the application and the threat model defined in WP1.

T5.7 Guidelines for third-party developers (ZIB:3, UCL:3, Stakk:1) This task will provide a document explaining to third-party developers how to develop self-managing applications using the software developed

in SELFMAN.

T5.8 Self-managing distributed theater application on mobile devices (UCL: 3) This task will build an application for mobile devices (specifically, gPhones and laptop computers) that demonstrates the self-managing transactional storage of WP3.

Deliverables

D5.1 Report on user requirements for application servers. **(M12)**

D5.2a Design specification of self-managing dynamic WWW server application. This deliverable reports on the design specification for both T5.2 and T5.3. **(M24)**

D5.2b Self-managing dynamic WWW server application for J2EE. **(M36)**

D5.3 Self-managing dynamic WWW server application for Mozart. **(M36)**

D5.4a Qualitative evaluation of self-management properties based on usage scenarios. **(M36)**

D5.4b Quantitative evaluation of self-management properties. **(M36)**

D5.6 Evaluation of security mechanisms. **(M36)**

D5.7 Guidelines for developing self-managing applications. **(M36)**

D5.8 Self-managing distributed theater application on mobile devices. **(M36)**

Milestones and expected result

M12: Understand requirements for application servers.

M24: Understand application structure for dynamic WWW server application.

M36: Finished dynamic WWW server applications.

M36: Understand effectiveness of self-* services for the application.

M36: Understand how to build self managing applications.

Workpackage 6 Description

Workpackage number	WP6		Start date or starting event:			0		
Workpackage title: Management, dissemination, and exploitation								
Participant id	1	2	3	4	5	6	7	8
Person-months per participant:	10	3	3	5	3	0	3	3

Objectives

To manage the project scientifically and administratively. To maximize the scientific progress. To disseminate the results, including as Open Source software. To collaborate with other projects.

Description of work

T6.1 Project management (UCL:7) This task will carry out the administrative management of the project, define the project standards and guidelines in relation to deliverables, presentations, and dissemination. The task will organize official project meetings and reviews. The task will coordinate, compile, and distribute project reports.

T6.2 Workpackage management (UCL:1, KTH:1, INRIA:1, FT R&D:3, ZIB:1, Stakk:1, NUS:1) This task will carry out the technical project management. The task will coordinate developments in the different workpackages. The task will ensure overall coordination of the project and resolve technical conflicts.

T6.3 Dissemination and exploitation (UCL:0.5, INRIA:0.5, FT R&D:1, Stakk:1, NUS:1) This task will implement a project website and Wiki with presentations, reports, publications, and deliverables. The task will organize project workshops. The task will explore potential applications of project results and potential partnerships with other projects and companies. This task will organize the Open Source dissemination of the project software. The ongoing work of this task (except for M36 which as D6.3) will be reported in the deliverables D6.5a, D6.5b, and D6.5c and the Periodic Activity Reports.

T6.4 Synergies and collaborations (UCL:0.5, INRIA:0.5, KTH:1, ZIB:1) This task will identify synergies and collaborations with other projects, old and new, in the IST portfolio. For description of this task, see the concertation clause in section 6.2. The work of this task will be reported in the deliverables D6.5a, D6.5b, and D6.5c and the Periodic Activity Reports.

T6.5 Assessment, evaluation, and lessons learned (UCL:1, KTH:1, INRIA:1, ZIB:1, FT R&D:1, Stakk:1, NUS:1) This task will evaluate the technical results of the project according to the assessment criteria for each workpackage. This task will give the lessons learned in the project, including general principles, insights gained, and the assessment of the use of two platforms and their comparison. The workpackage coordinators will report to the project coordinators. The evaluation will be done at months 12, 24, and the final evaluation at month 36 according to the measurable objectives given in section 6.1.3.

Deliverables

D6.1a Project website and Wiki. **(M1, M6, then continuously updated)**

D6.1b First project workshop. **(M12)**

D6.1c Second project workshop. **(M24)**

D6.1d Third project workshop. **(M36)**

D6.3 Dissemination and use report. **(M36)**

D6.5a First progress and assessment report with lessons learned. **(M12)**

D6.5b Second progress and assessment report with lessons learned. **(M24)**

D6.5c Final progress and assessment report with lessons learned. **(M36)**

Milestones and expected result

M12: Project making good progress (all deadlines respected).

M24: Possible synergies and collaborations are realized.

M36: Understand general principles of self management.

M36: Understand effectiveness of two platform implementation.

8. Project Resources and Budget Overview

8.1 Efforts for the Full Duration of the Project

STREP/STIP Effort Form - Full duration of project
Project number (acronym): 34084 (SELFMAN)

STREP/STIP Activity type	UCL	KTH	INRIA	FT R&D	ZIB	E-Plus	NUS	Stakk	TOTAL ACTIVITIES
RTD/Innovation activities									
WP1 Structured overlay network and basic mechanisms	14	10	8	0	0	0	17	2	51
WP2 Service architecture and component model	17	12	19	11	11	0	0	0	70
WP3 Self-managing storage and transactions	6	11	0	8	15	0	0	3	43
WP4 Self-management services	6	14	19	3	15	0	17	2	76
WP5 Application requirements and evaluations	9	5	6	9	11	0	3	4	47
WP6 Management, dissemination, and exploitation (Tasks T6.3, T6.4, T6.5)	2	2	2	2	2	0	2	2	14
Total research/innovation	54	54	54	33	54	0	39	13	301

Consortium management activities	UCL	KTH	INRIA	FT R&D	ZIB	E-Plus	NUS	Stakk	TOTAL ACTIVITIES
WP6 Management, dissemination, and exploitation (Tasks T6.1 and T6.2)	8	1	1	3	1	0	1	1	16
Total consortium management	8	1	1	3	1	0	1	1	16

	UCL	KTH	INRIA	FT R&D	ZIB	E-Plus	NUS	Stakk	
TOTAL per Participant	62	55	55	36	55	0	40	14	

Overall TOTAL EFFORTS									317

8.2 Overall Budget for the Full Duration of the Project

Contract Preparation Forms

EUROPEAN COMMISSION
6th Framework Programme on Research, Technological Development and Demonstration

Specific Targeted Research or Innovation Project

A3.1

Please use as many copies of form A3.1 as necessary for the number of partners

Proposal Number	034084	Proposal Acronym	SELFMAN

Financial information - whole duration of the project

Partici pant nº	Organisation short name	Cost model used	Estimated eligible costs and requested EC contribution (whole duration of the project)		Costs and EC contribution per type of activities			Total (4)=(1)+(2)+ (3)	Total receipts
					RTD or innovation related activities (1)	Demonstration activities (2)	Consortium Management activities (3)		
1	UCL	AC	Eligible costs	Direct Costs (a)	301,975.00		47,090.00	349,065.00	
				of which subcontracting				.00	
				Indirect costs (b)	60,395.00		8,418.00	68,813.00	
				Total eligible costs (a)+(b)	362,370.00	.00	55,508.00	417,878.00	
			Requested EC contribution		362,370.00		55,508.00	417,878.00	
2	KTH	AC	Eligible costs	Direct Costs (a)	331,000.00		10,800.00	341,800.00	
				of which subcontracting				.00	
				Indirect costs (b)	66,200.00			66,200.00	
				Total eligible costs (a)+(b)	397,200.00	.00	10,800.00	408,000.00	
			Requested EC contribution		397,200.00		10,800.00	408,000.00	
3	INRIA	FC	Eligible costs	Direct Costs (a)	270,018.00		3,742.00	273,760.00	
				of which subcontracting				.00	
				Indirect costs (b)	261,618.00		4,625.00	266,243.00	
				Total eligible costs (a)+(b)	531,636.00	.00	8,367.00	540,003.00	
			Requested EC contribution		265,818.00		8,367.00	274,185.00	
4	FT	FC	Eligible costs	Direct Costs (a)	305,399.00		20,789.00	326,188.00	
				of which subcontracting				.00	
				Indirect costs (b)	207,483.00		5,342.00	212,825.00	
				Total eligible costs (a)+(b)	512,882.00	.00	26,131.00	539,013.00	
			Requested EC contribution		256,441.00		26,131.00	282,572.00	
5	ZIB	AC	Eligible costs	Direct Costs (a)	280,015.00		8,056.00	288,071.00	
				of which subcontracting				.00	
				Indirect costs (b)	56,003.00		1,611.00	57,614.00	
				Total eligible costs (a)+(b)	336,018.00	.00	9,667.00	345,685.00	
			Requested EC contribution		336,018.00		9,667.00	345,685.00	
7	NUS	AC	Eligible costs	Direct Costs (a)	104,774.00		5,454.00	110,228.00	
				of which subcontracting				.00	
				Indirect costs (b)	18,489.00		963.00	19,452.00	
				Total eligible costs (a)+(b)	123,263.00	.00	6,417.00	129,680.00	
			Requested EC contribution		123,263.00		6,417.00	129,680.00	
8	Stakk	FC	Eligible costs	Direct Costs (a)	147,200.00		8,000.00	155,200.00	
				of which subcontracting				.00	
				Indirect costs (b)	36,800.00		2,000.00	38,800.00	
				Total eligible costs (a)+(b)	184,000.00	.00	10,000.00	194,000.00	
			Requested EC contribution		92,000.00		10,000.00	102,000.00	
TOTAL			Eligible costs		2,447,369.00	.00	126,890.00	2,574,259.00	.00
			Requested EC contribution		1,833,110.00	.00	126,890.00	1,960,000.00	

Contract Preparation Forms

**Specific Targeted
Research or Innovation
Project**

A3.2

| Proposal Number | 034084 | | Proposal Acronym | SELFMAN |

Estimated breakdown of the EC contribution per reporting period				
Reporting Periods	**Start month**	**End month**	**Estimated Grant to the Budget**	
			Total	**In which first six months**
Reporting Period 1	1	12	560,000.00	.00
Reporting Period 2	13	24	830,000.00	410,000.00
Reporting Period 3	25	36	570,000.00	280,000.00
Reporting Period 4			.00	.00
Reporting Period 5			.00	.00
Reporting Period 6			.00	.00
Reporting Period 7			.00	.00

8.3 Management Level Description of Resources and Budget

The following table gives an overview of the budget utilisation for each partner in Euros. The numbers in this table are given as orientative for how each partner will split his/her budget. For each partner, we give the rate (Euros/month) for the personnel, the travel and subsistence budget, and the equipment budget. All these numbers are for the duration of the project. The rates, travel and subsistence budget, and equipment budget are calculated with respect to the numbers in the "Requested" column of the table. For example, for INRIA this means that we have 4367*54 + 20000 + 10000 = 265818.

Partner	Model	RTD/Innovation activities						Management activities			Total Req.
		Rate/PM (incl OH)	PM	Travel +subs.	Equip.	Costs	Req.	PM	Costs	Req.	
UCL	AC	6080	54	21000	13050	362370	362370	8	55508	55508	417878
KTH	AC	6800	54	20000	10000	397200	397200	1	10800	10800	408000
INRIA	FC	4367	54	20000	10000	531636	265818	1	8367	8367	274185
FT R&D	FC	7377	33	9000	4000	512882	256441	3	26131	26131	282572
ZIB	AC	5667	54	20000	10000	336018	336018	1	9667	9667	345685
E-Plus	FC	6000	0	0	0	0	0	0	0	0	0
NUS	AC	2417	39	20000	9000	123263	123263	1	6417	6417	129680
Stakk	FC	6000	13	10000	4000	184000	92000	1	10000	10000	102000
Total						2447369	1833110		126890	126890	1960000

Audit costs of 4000 per partner have been included in management costs. For UCL, the management costs also include 2868 equipment costs for management equipment.

8.3.1 AC Partners "Own" Contributions

AC Partner	Names of persons	Total PM	Euros/ PM	Total
UCL	Peter Van Roy, Stéphanie Landrain	14.4	4500	64800
KTH	Seif Haridi	7.2	10300	74160
ZIB	Alexander Reinefeld	7.2	8500	61200
NUS	Roland Yap	7.2	6500	46800

Each AC partner will dedicate a part of its permanent staff to the project. **UCL**. Peter Van Roy (professor) and Stéphanie Landrain (administrative assistant) will dedicate 20% of their time to the SELFMAN project. In addition, we will use the university resources (the staff of the personnel service, the financial service, the research administration) to help with the cost statements, the contract preparations, and the ongoing project management. **KTH**. KTH will contribute 20% of a senior staff member to the project. **ZIB**. ZIB will contribute 20% of a senior staff member to the project. **NUS**. NUS will contribute 20% of the time of a senior staff member to the project.

9. Ethical Issues

Table A. Proposers are requested to fill in the following table:

Does your proposed research raise sensitive ethical questions related to:	YES	NO
Human beings		X
Human biological samples		X
Personal data (whether identified by name or not)		X
Genetic information		X
Animals		X

Table B. Proposers are requested to confirm that the proposed research does not involve:

Research activity aimed at human cloning for reproductive purposes,

Research activity intended to modify the genetic heritage of human beings which could make such changes heritable[1]

Research activity intended to create human embryos solely for the purpose of research or for the purpose of stem cell procurement, including by means of somatic cell nuclear transfer.

	YES	NO
Confirmation: the proposed research involves none of the issues listed in Table B	X	

---

[1] Research relating to cancer treatment of the gonads can be financed

10. Other Issues

10.1 Gender Issues

The SELFMAN project is gender-neutral. We will hire the best people independent of their gender. Nevertheless, we recognize the underrepresentation of women in computer science research. This is a perennial problem that has no recognized solution. For SELFMAN, we will encourage women to apply to the project and given equal qualifications we will give preference to woman candidate researchers.
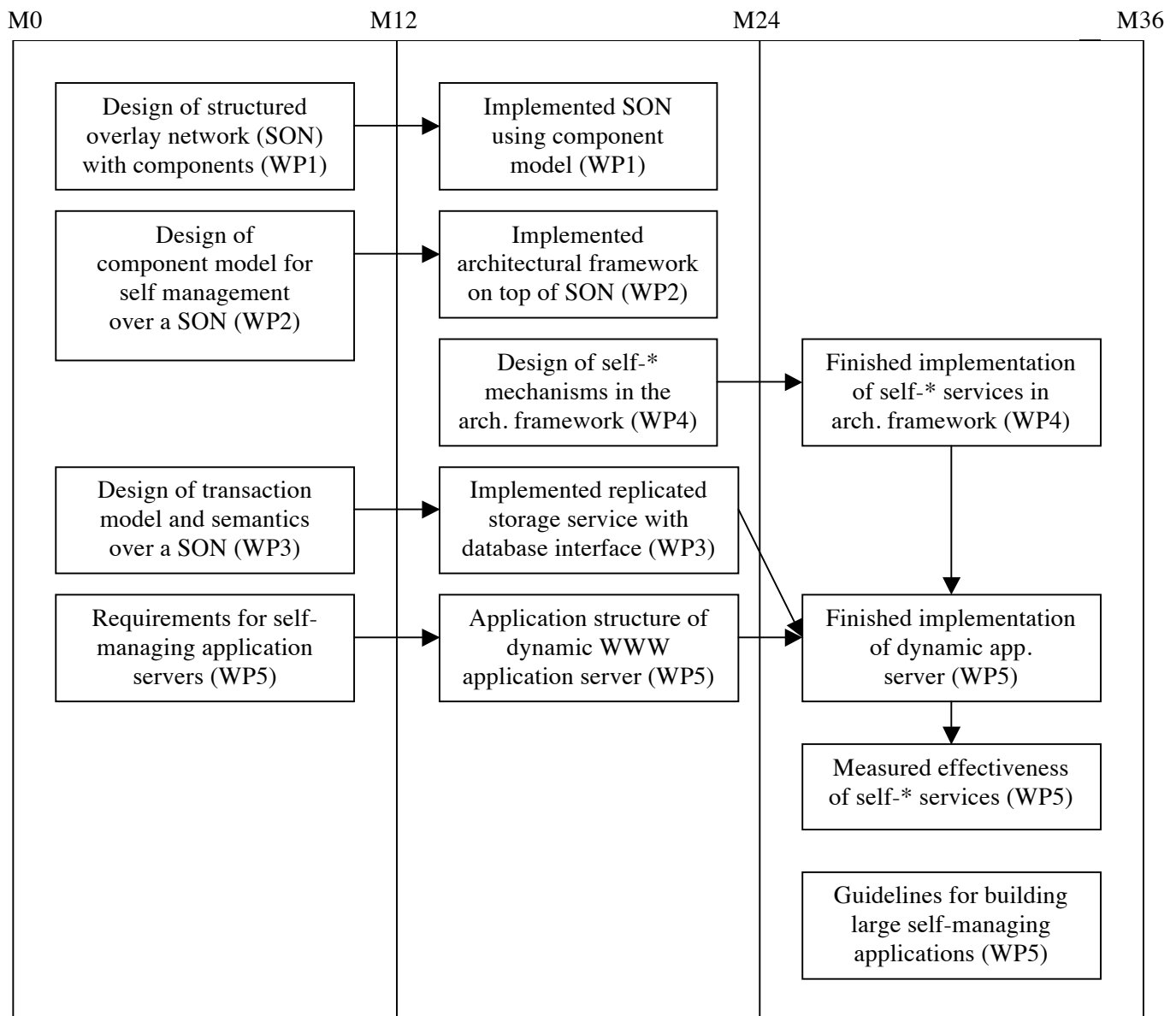
10.2 Policy Issues

The SELFMAN project will facilitate increased use of the Internet by making it easier to construct self-managing applications. In addition, we will encourage dissemination of the software results of the project as Open Source software. Both of these activities taken together give us the expectation that the SELFMAN project will increase the use of the Internet in all layers of society, including the Third World and other entities that do not have the material resources to do systems management. Self management, by its very nature, is much less expensive than human management. With our software, one of the goals of SELFMAN is that self management should also not require advanced technical knowledge.

Appendix A – Consortium Description

A.1 Project Roadmap

→ = main dependencies between achievements in roadmap

M0   M12   M24   M36

Design of structured overlay network (SON) with components (WP1)	→	Implemented SON using component model (WP1)		

| Design of component model for self management over a SON (WP2) | → | Implemented architectural framework on top of SON (WP2) | | |

| | | Design of self-* mechanisms in the arch. framework (WP4) | → | Finished implementation of self-* services in arch. framework (WP4) |

| Design of transaction model and semantics over a SON (WP3) | → | Implemented replicated storage service with database interface (WP3) | | |

| Requirements for self-managing application servers (WP5) | → | Application structure of dynamic WWW application server (WP5) | → | Finished implementation of dynamic app. server (WP5) |

| | | | | Measured effectiveness of self-* services (WP5) |

| | | | | Guidelines for building large self-managing applications (WP5) |

A.2 Participants and Consortium

The SELFMAN project requires four special areas of scientific expertise, namely structured overlay networks, component models, transactional storage, and programming languages. In addition, we require technological expertise in the J2EE model and we require industrial usage scenarios and trace data. We now present a table with the partners' expertise in these areas and show their complementarity.

Partner	Background	Role	Foreground (see also section 6.1.4)
UCL	P2PKit architecture and P2PS library (from PEPITO, EVERGROW), Mozart component model, Mozart system	Expertise in structured overlay networks and Mozart implementation	Component model for structured overlays, Mozart implementation
KTH	DKS protocol and middleware (from PEPITO, EVERGROW), J2EE and Grid implementation, Mozart system	Expertise in structured overlay networks and J2EE implementation	Structured overlay for component model, J2EE implementation
INRIA	Fractal model (from SARDES), data replication management (from GORDA), self management for component models	Expertise in component models and self management	Service architecture for self management, self management services
FT R&D	Fractal model (from SARDES), use and management requirements for application hosting, traffic generation	Expertise in component models, use scenarios, and traffic generation	Application requirements and evaluation
ZIB	ZIB Data Management System (from ZIBDMS), Grid implementation	Expertise in storage services and Grid	Storage service
E-Plus	Use and management requirements for application hosting, trace data, low-level self management (from FLEXINET)	Expertise in use scenarios and trace data	Application requirements and evaluation
NUS	Security auditing and monitoring	Expertise in security	Security requirements, mechanisms, and evaluation
Stakk	Media streaming technologies	Expertise in media streaming scenarios and trace data	Application requirements and evaluation

We now explain the content of this table in more detail. KTH and UCL provide expertise in structured overlay networks; they were both project partners in the PEPITO project, which developed this area. KTH has developed the DKS protocol and a Java middleware also called DKS based on this protocol [DKS05, GHOD05]. UCL has developed a Mozart library, P2PS [P2PS05], based on the Tango protocol, which is a more scalable variant of the DKS protocol [CART05]. In addition, UCL has started work on a service architecture, P2PKit [P2PK05], which is able to run applications on top of P2PS. This preliminary work will be the starting point for WP1. INRIA and France Telecom R&D have jointly developed an advanced component model, the Fractal model [BRUN04]. This model will be the starting point for WP2. ZIB is developing a data management system for distributed data that will run on a peer-to-peer base [SCHU04, SCHU05]. This system will be the starting point for WP3 and a basis for the development of the trading services in WP4. UCL has also designed a transaction model for structured overlay networks [MESA05]. INRIA is working on self management as part of its work on component models in the SARDES project, and has developed a first architecture for self managing cluster-size systems [BOUC05]. This work will be the starting point for WP4, which will extend it to self management over the service architecture. France Telecom and INRIA have an ongoing cooperation on distributed configuration management whose results will be a starting point for the self-configuration task in WP4. NUS brings valuable experience in security and in programming languages to the project; since they have already collaborated with

KTH this will facilitate their integration into the project. Stakk is a start-up company working on a media streaming application. This work will give us realistic usage scenarios and trace data for evaluating the self-management services.

Université catholique de Louvain (UCL)

The Catholic University of Louvain, founded in 1425, was bifurcated in 1971, giving birth to two independent entities, UCL and its sister university KUL (Katholieke Universiteit Leuven). While the KUL remained in Louvain, UCL established itself in the new town of Louvain-la-Neuve. UCL currently has more than 20000 students (including 4000 foreign students), 3000 teaching and research personnel, and 1800 technical and administrative personnel. UCL hosts more than 1000 external research contracts with an annual turnover of 70 million Euros and is the nucleus of an industrial park containing almost 100 companies. UCL is an active participant in the ERASMUS (now SOCRATES) and TEMPUS programs, and is a founding member of MED-CAMPUS, the Coimbra Group, CLUSTER, FIUC, NATURA, CEMS, etc.

The Department of Computing Science and Engineering at UCL (INGI) is part of the Faculty of Applied Science. INGI has around 50 teaching and research personnel, with a twofold mission: education (offering engineering and doctoral degrees) and research. The department is recognized worldwide for its contributions to dependable distributed computing, networking, constraint and logic programming, programming languages, and software engineering. Major releases by the department include the KAOS methodology and its tools for requirements engineering, and the Mozart Programming System for distributed application development and constraint programming. INGI collaborates closely with research groups in many areas of the world including the USA, France, Germany, Sweden, and the third world. INGI personnel are major contributors and catalyzers of computer technology and applications in Belgium. For example, INGI is a key player in the creation and continuing development of the CEDITI s.a., a software consulting and services company specializing in multimedia and Web development and software engineering, in the WIN, the Wallonia High-performance Intranet. Current INGI projects include EVERGROW (European IP), CoreGrid and E-Next (European NoE), MILOS, TOTEM, BioMaze, TransMaze, ReQuest, APPAREIL (funded by Walloon Region of Belgium), and many smaller projects.

UCL key personnel

Peter Van Roy is professor in the Dept. of Computing Science and Engineering at UCL, where he heads the research group on Programming Languages and Distributed Computing. He holds a Ph.D. from the University of California at Berkeley and a Habilitation à Diriger des Recherches from the Université Paris VII. He has been involved in several European projects including ACCLAIM, PEPITO, and EVERGROW, as well as the regional projects PIRATES and MILOS. He has been on the program committee and invited speaker for numerous international conferences. He was a member of the International Scientific Council of the IRCICA research institute in Lille, France. He was the first Belgian Executive Committee representative when Belgium joined ERCIM in 2004. He developed Aquarius Prolog, the first Prolog compiler to generate code competitive in performance with C compilers. He holds one patent in graphic design and developed the commercial Macintosh application FractaSketch based on this patent. He is a developer of the Mozart Programming System, an advanced platform for distributed intelligent applications, and is currently a member of the Mozart Board. He is codesigner (with Seif Haridi and Per Brand) of the distribution model of Mozart. With Seif Haridi he has written a comprehensive textbook, "Concepts, Techniques, and Models of Computer Programming", that was published by MIT Press in 2004. He is currently working on tools and techniques for simplifying the development of robust collaborative applications on the Internet. In the PEPITO project he worked on structured overlay networks and started the design of an architecture for building applications on top of them. This work is continuing in the EVERGROW project and is using the PlanetLab infrastructure for testing and deployment purposes. The SELFMAN project will greatly extend and expand this work into the area of self management.

Royal Institute of Technology (Kungliga Tekniska Högskolan – KTH)

KTH is one of the major engineering schools in Sweden. KTH will participate through the Laboratory of Electronics and Computer Systems (LECS) in the IMIT department. The activities of LECS concern Computer Systems at large, but in particular parallel computer systems, computer architecture, and distributed computer systems including programming systems. The research is mainly motivated by technical challenges in distributed and parallel systems. In all of our projects, we focus as well on experimental design and evaluation as on sound design principles.

Prof. Haridi's group in LECS co-develops the programming system Mozart for transparent programming of distributed and mobile applications. The system greatly simplifies the development of net-based services. The research is now focused on aspects such as scalability, security and support for fault tolerance (in cooperation with SICS and Ericsson). There is also work on adaptation of Mozart to small mobile units (PDAs) and algorithms for distributed systems and fault-tolerant systems, in particular work on self-stabilizing algorithms and also program analysis of parallel and concurrent systems.

In 1994, LECS researchers developed an extensible object-oriented platform NUTS for distributed computing. The language of NUTS is a concurrent object-oriented programming language with coarse-grained parallelism and distributed shared memory model implemented on a distributed memory architecture, e. g., a network of workstations. NUTS processes can be arranged into structured collections: grids that enable to program data-parallel computations on a high level. The distributed layer of NUTS supports passing objects, classes and scripts among distributed NUTS processes, and is based on serialization technique similar to the Java object serialization developed later at SUN Microsystems.

KTH key personnel

Seif Haridi is professor at KTH of the computer systems chair and Chief Scientist of SICS. He is currently involved on research in the area of Peer-to-Peer overlay computing, high availability, and large-scale agent-based simulations. Haridi is the scientific coordinator of the EU project PEPITO (see http://www.sics.se/pepito) that finished in June 2005, a project devoted to Peer-to-Peer computing. He is the scientific coordinator of the EU/FET project EVERGROW (see http://www.evergrow.org). Previously he led the subproject on research in Peer-to-Peer computing in EVERGROW. His research group has designed DKS, an architecture based on structured Distributed Hash Table overlay networks for large-scale distributed applications. Demonstrator applications done with DKS include including global file systems, media distribution, and GRID computing applications (see http://dks.sics.se). Already the architecture has been used for mobile weblogs with Ericsson, and as the reference architecture for a project with Swedish Defense on Network-based Defense. The research group is also designing a language-independent middleware for distributed and peer-to-peer computing based on the experience of the Mozart system developed by my research group together with German and Belgian researchers. We have also developed a large-scale agent based simulation in the EU project ICITIES on multiprocessor clusters. Haridi is also a co-designer of the programming language Oz and the Mozart programming platform (see http://www.mozart-oz.org). Earlier research includes implementation of logic and constraint-based languages, and scalable cache-coherent parallel computers. He led the development of SICStus Prolog, a high-quality Prolog system that is the most widely used worldwide for education and research on UNIX workstations. He was a team member of the Aurora project generating the first parallel Prolog on shared memory machines. He was a co-leader of the Andorra Kernel Language (AKL) team and co-designer of the concepts of AKL, the first existing complete concurrent constraint language. He is a co-inventor of COMA architectures, a scalable cache-coherent multiprocessor with only caches. This concept has been taken by SUN Microsystems (see publication list). One important recent work is the book published by MIT-Press: Concepts Techniques and Models of Computer Programming (http://www.info.ucl.ac.be/people/PVR/book.html) The book is used in several universities for teaching computer programming, and is considered by many as the main 'bible' in the area.

Institut National de Recherche en Informatique et Automatique (INRIA)

INRIA (National Institute for Research in Computer Science and Control) is a French public-sector scientific and technological institute operating under the dual authority of the Ministry of Research and the Ministry of Industry. INRIA's missions are "to undertake basic and applied research, to design experimental systems, to ensure technology and knowledge transfer, to organize international scientific exchanges, to carry out scientific assessments, and to contribute to standardization" The research carried out at INRIA brings together experts from the fields of computer science and applied mathematics covering the following areas: Networks and Systems; Software Engineering and Symbolic Computing; Man-Machine Interaction; Image Processing, Data Management, Knowledge Systems, Simulation and Optimization of Complex Systems.

INRIA's ambition is to be a world player, a research institute at the heart of the information society. INRIA aims to network skills and talents from the fields of information and computer science and technology from the entire French research system. This network allows scientific excellence to be used for technological progress, for creating employment and wealth and for new uses in response to socio-economic needs. INRIA's decentralized organization (6 Research Units), its small autonomous teams, and regular evaluation enable INRIA to develop its partnerships, with 95 research projects shared with universities, grandes Ecoles and research organizations. It is also strengthening its involvement in the development of research results and technology transfer.

INRIA gathers in its premises around 3000 persons: 900 INRIA permanent employees (400 researchers, 500 engineers and technicians), 750 post-docs, engineers and visitors, 700 doctoral candidates, 450 Researchers and professors from other organizations and 200 "Expert engineers" (on research contracts). The Institute has extensive collaborations with industrial partners. 600 contracts with industrial partners are currently active, 40% of them being European funded ones. In addition, the Institute is very active in promoting high-tech start-up companies through its INRIA transfer and I-Source subsidiaries. INRIA is a member of ERCIM EEIG, European Research Consortium for Computer Science and Mathematics. Outside Europe, INRIA also has a significant activity: it has created joint research laboratories (Russia and China), signed cooperation agreements (NSF, India, Brazil, etc.) and promotes intensive scientific exchanges.   INRIA Web: http://www.inria.fr/

INRIA key personnel

Jean-Bernard Stefani is a Research Director at INRIA, where he leads the Sardes project. From 1990 to 2000, he was Head of the Distributed Systems research laboratory at France Telecom R&D. During that period, he was also Chairman of the Working Party in ITU-T Study Group 7, responsible for the development of open distributed processing standards. He has been involved in several Esprit, ACTS and IST projects, including Esprit ISA, ACTS RetINA, and IST Mikado. He is one of the initiators and a past Chairman of the Board of the ObjectWeb Consortium. He has served on the program committee of several international conferences (Middleware, DOA, Forte/PSTV, FMOODS, SRDS, etc). His current research interests include: dynamically configurable distributed systems, formal models for component-based distributed programming, large-scale distributed systems monitoring and management.

Noel de Palma is an Assistant Professor at Institut National Polytechnique de Grenoble (INPG) since September 2002, and a member of the Sardes project. From 2001 to 2002, he has been a postdoctoral researcher at France Telecom R&D. His main research interests include distributed system construction, configuration management, autonomous distributed management, and component-based systems. Noel de Palma holds a PhD from INPG.

Alan Schmitt is a Researcher at INRIA since January 2004, and a member of the Sardes project. From September 2002 to January 2004, he has been a postdoctoral researcher at the University of Pennsylvania, USA. His main research interests cover distributed, component-based, and functional programming, programming language semantics, type systems, and concurrency theory. Alan Schmitt holds a PhD from Ecole Polytechnique.

France Telecom Research & Development (FT R&D)

France Telecom R&D is Europe's leading telecommunications R&D center, with areas of research including human-machine interactions, mobility, network architecture, fixed/mobile/Internet convergence, and very high throughput transmission and access networks. With more than 3,400 engineers, researchers and technicians spread in 14 sites in Europe, North America and Asia, France Telecom R&D is implemented at international level that enables cooperation with influent industrial groups, the international scientific community and regulatory bodies. Responsible for the Group's strategic targets and critical technical areas,

France Telecom R&D has been contributing to numerous collaborative R&D actions, both at national and international level: RNRT (National Research Network in Telecommunications), IST projects, EUREKA, etc. In addition, France Telecom Research & Development maintains a close relationship with public-sector laboratories, such as the French National Center for Scientific Research (CNRS), the INRIA and the CEA. It is also a founding member of the ObjectWeb Consortium (http://www.objectweb.org/).

France Telecom R&D contributes to SELFMAN through its Software Techniques and Engineering division, specifically the laboratory MAPS/AMS, which is an active contributor in the ObjectWeb Consortium. Research projects cover topics such as the design and development of flexible distributed object-oriented platforms and component-based systems. In particular, France Telecom R&D developed the Fractal model conjointly with the Sardes INRIA team. Fractal is a modular and extensible component model that can be used to design, implement, deploy, and reconfigure various systems and applications, from operating systems to middleware platforms), persistence and transaction frameworks (Jorm, Medor and Speedo), generic load injection platform for benchmarking middleware (Cliff), and applications of distributed systems, in particular, the design of enterprise information systems.

FT R&D key personnel

Thierry Coupaye heads the Distributed Software Architectures & Infrastructures Research Pole in the France Telecom R&D Division. He completed his Ph.D. in Computer Science from the UJF Grenoble University, France, in 1996 in the area of active databases (Event-Condition-Action rules) and worked afterwards as a teaching and research assistant at INPG Technological University. Then he worked as a researcher at the European Bioinformatics Institute (EMBL-EBI) in Cambridge, U.K., in the area of semi-structured data management for genomics, and then in the Dassault Systems and University of Grenoble Joint Laboratory where he worked on large-scale software deployment. He joined France Telecom in 2000. He led several R&D projects in the database and software architecture areas and then took lead of the Distributed Software Architectures & Infrastructures Research Pole in 2003. He has been involved in several collaborative projects (Esprit Goodstep, ITEA Osmose, IST Artist). He is the author of more than 30 refereed articles and has participated in several program and organization committees of conferences in these areas (IDEAS, ETAPS, Euromicro, etc.). His current research interests include software architecture, component-based systems, aspect-oriented programming, reflexive systems, and autonomic computing.

Bruno Dillenseger has been working for France Telecom R&D in the field of object-oriented distributed systems and communication middleware for more than 13 years. His main contributions focus on mobile and intelligent agent technology, with a number of platforms and publications. Three years ago, he launched the CLIF project in the context of ObjectWeb open source consortium's JMOB initiative (Java Middleware Open Benchmarking), to provide the benchmarking community with a component-based distributed framework for generating load on, and measuring performance of, arbitrary systems (see CLIF project at http://clif.objectweb.org/). One year ago, he got involved in autonomic computing related research from a software architecture and performance perspective.

Nicolas Rivierre completed his Ph.D. in Computer Science from the Paris VI University, France, in 1998 in the area of real-time scheduling. He joined INRIA in 1992, working as a research engineer in the area of real-time and fault tolerant

systems. He joined France Telecom in 1997, working in several R&D projects in the area of Distributed Software Architectures & Infrastructures. His research interests include software component-based systems, system management and formal methods.

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

The Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) is a non-university research institute of the State of Berlin for applied mathematics and computer science. Within the CS department, which is headed by Prof. Alexander Reinefeld, ZIB operates high-performance computers that are among the most powerful in Germany. As a major driving force in science and science-politics, ZIB pushed the establishment of the German E-Science initiative D-Grid, the GGF and E-Grid. ZIB has demonstrated its excellence in Grid research in the EU projects DataGrid, GridLab, FlowGrid, CoreGrid, and GridCoord.

Our research focuses on parallel and distributed systems, more specifically on distributed data management in Grids, scalable services, and autonomous computing. With providing consulting services to high performance computing users and developing parallel and distributed systems for clusters and Grids, we are in the unique position to be able to evaluate designs from both perspectives, from the HPC users' point of view and from the point of view of Grid researchers.

ZIB key personnel

Alexander Reinefeld heads the computer science department at Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) and holds a chair for Parallel and Distributed Systems at the Berlin Humboldt University. He co-founded the European Grid Forum and the Global Grid Forum, where he is a member of the Advisory Committee (GFAC). He was one of the major driving forces in the establishment of the German E-Science initiative D-Grid, which got funded by the Federal Ministry of Education and Research (BMBF) with more than 100 million Euros. Alexander Reinefeld acts as an advisor at a national and international level – e.g. for the 7th FP and the EU NGG and NGG-2 group. He organized international conferences on Grid computing like CCGrid in 2002 and GGF in 2004, and co-organized several workshops on Grid and Cluster computing. He participates in editorial boards (FGCS, JoGC, IJGUC) and published numerous scientific papers and some books.

National University of Singapore (NUS)

NUS (National University of Singapore) is the premier university in Singapore and is acknowledged as one of the finest universities in the Asia-Pacific region. In the 2004 global survey of universities by the Times of London, NUS was placed among the world's 20 top universities. The NUS mission is to provide quality education with a broad-based curriculum and engage in high impact research. There are approximately 30,000 students split into 22,000 undergraduate students and 8,000 graduate students. NUS has a strong research culture and close tie-up between teaching and research with 13 national-level, 12 university-level and more than 60 faculty-based research centers.

There are 13 faculties and schools at NUS, which includes Engineering, Computing, Science, Arts and Social Sciences, Business, Law, Medicine, Dentistry, Public Policy and Design and Environment, and the School of Computing. The School of Computing encompasses the field of information technology and comprises of two departments: the Department of Computer Science, and the Department of Information Systems. The former focuses on the fundamental and technical aspects of computer science and technology, while the latter is centered on combining aspects of computing

with practical applications in management. The School of Computing has approximately 2,000 students, which includes 500 graduate students.

The Department of Computer Science has strong research groups in the areas of Programming Languages and Software Engineering, Systems, Networking, Databases, Artificial Intelligence, Computational Biology and Multimedia. In the 2004/2005 period, the department received close to SGD$10M of external funding for research. The overall research profile has high impact and this can be seen in the citation ranking under the ISI Web of Knowledge where the Department of Computer Science, NUS, is ranked 33 globally.

NUS key personnel

Roland Yap Hock Chuan is an associate professor at the National University of Singapore in the Department of Computer Science, School of Computing. He is well known as one of the primary authors of the CLP(R) system, which is the first Constraint Logic Programming (CLP) language and system. His work on CLP(R) has been influential in programming language research and also constraint systems. He has served on the program committee of numerous international conferences. Some recent conferences are: PADL06, ICLP06, PRICAI06, CP05, IJCAI05, and SAC05. His current research interests include: concurrent and distributed computing, constraint programming, systems security, and software engineering. In the area of systems security, he has an existing research funded by the Defence Science Technology Agency, Singapore, which focuses on application and operating systems infrastructure for increasing security and trust in Unix and Windows. He has worked on auditing and monitoring (including intrusion detection) mechanisms, at both high and low levels, for secure systems. One of the issues is that the monitoring mechanism itself is subject to attack. This work will be extended in SELFMAN to become a self monitoring infrastructure built on top of a structured overlay network.

Stakk AB

Stakk AB is a Swedish company offering distribution solutions for online video and other media. Stakk's objective is to establish a leading distribution network for real time streaming and for on demand content by providing high quality services at significantly lower costs.

Stakk's uses a p2p solution to distribute online video. The p2p solution is easily embedded in services and hardware for internet video and IPTV, making consumer applications and appliances part of Stakk's or the customer's own p2p-based network. By embedding its software, Stakk provides built-in, high quality video distribution capabilities to other products. The benefits can be made available to many participants in the value chain including consumers and companies that already make use of video streaming services.

Stakk works closely with the p2p group at the Distributed Systems Lab at the Swedish Institute for Computer Science (SICS) and their collaborators at KTH, in Stockholm. SICS, KTH, and the Stakk research team have provided the foundation for building a p2p communication network for streaming of video that is robust, highly scaleable and to a large degree self-managed.

Stakk key personnel

Mohammed El-Beltagy has extensive experience applying optimization, machine learning, simulation, modeling and agent-based models to a variety of business and engineering problems. He has a Bachelor's degree in Mechanical engineering design from the American University in Cairo, a Master's degree in Mechatronics from Lancaster University, and a PhD in Mechanical Engineering from the University of Southampton, England. After obtaining his PhD degree, he

honed his consulting expertise as Senior Scientist at BiosGroup, Inc., where he helped optimize various aspects of the operations of Fortune 50 companies. Later he returned to Egypt where he became the principal of Optomatica, an analytic business consulting concern. He holds an assistant professor position at Cairo University where he lectures on Computational Intelligence, Game Theory and Network Optimization.

Sameh El-Ansary is a well-recognized researcher in the area of large-scale distributed systems with emphasis on peer-to-peer structured overlay networks. He has a Bachelor's and a Master's degree in Computer Science from the American University in Cairo, and a PhD in the same discipline from the Royal Institute of Technology in Stockholm. Sameh El-Ansary has often sought business applications for his ideas. He was the winner of the PROGKOM2 Contest for the best ideas with commercialization potential stemming from research. He was also part-time software architecture consultant for the largest IT development company in Egypt (ITWorx). El-Ansary is currently an assistant professor at Nile University. He lectures on distributed algorithms, peer-to-peer systems, and network programming.

Sameh El-Ansary and Mohammed El-Beltagy work together at Stakk, combining expertise in distributed computing and optimization to seek a novel approach to peer-to-peer video streaming.

A.3 Sub-contracting

The project will do no sub-contracting.

A.4 Third Parties

None of the project work will be carried out by third parties.

A.5 Funding of Third Country Participants

The National University of Singapore (NUS) in Singapore is included as a project partner. We will use the expertise of the NUS in security and in programming language and system design. The key person at NUS who we will collaborate with is Roland Yap. Several project partners have a long history of fruitful collaboration with Roland Yap and with other researchers at NUS. NUS will contribute at a level of 40 person-months over the whole project. We expect to have one project meeting in Singapore over the duration of the project.

References

[ABER04] K. Aberer, A. Datta, and M. Hauswirth. Efficient, Self-Contained Handling of Identity in Peer-to-Peer Systems. IEEE Transactions on Knowledge and Data Engineering 16(7), July 2004, pages 858-869.

[ABER05] K. Aberer, L. Onana Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi. The essence of P2P: A reference architecture for overlay. In *5th International Conference on Peer-to-Peer Computing (P2P 05)*, IEEE Computer Society, 2005.

[ACHE02] F. Acherman. Forms, Agents, and Channels. PhD dissertation, U. of Bern, Switzerland, 2002.

[ALDR02] J. Aldrich and C. Chambers and D. Notkin. Architectural Reasoning in ArchJava. Proceedings 16th European Conference on Object-Oriented Programming (ECOOP), 2002.

[ALDR03] J. Aldrich and V. Sazawal and C. Chambers and David Notkin. Language Support for Connector Abstractions. Proceedings 17th European Conference on Object-Oriented Programming (ECOOP), 2003.

[ALLE97] R. Allen and D. Garlan. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, 6(3), 1997.

[AURE04] Erik Aurell and Sameh El-Ansary. A physics-style approach to scalability of distributed systems. SICS Technical Report T2004:01, 2004.

[BAUS02] W. Bausch, C. Pautasso, R Schaeppi, and G. Alonso. BioOpera: Cluster-Aware Computing. In Proc. IEEE International Conference on Cluster Computing (CLUSTER 2002). IEEE Computer Society, 2002.

[BENS01] I. Ben-Shaul and O. Holder and B. Lavva. Dynamic Adaptation and Deployment of Distributed Components in Hadas. IEEE Transactions on Software Engineering, 27(9), 2001.

[BLAI01] G. Blair and G. Coulson and A. Andersen and L. Blair and M. Clarke and F. Costa and H. Duran-Limon and T. Fitzpatrick and L. Johnston and R. Moreira and N. Parlavantzas and K. Saikoski. The Design and Implementation of OpenORB v2. IEEE Distributed Systems Online, 2(6), Special Issue on Reflective Middleware, 2001.

[BOUC05] S. Bouchenak, F. Boyer, D. Hagimont, S. Krakowiak, A. Mos, N. de Palma, V. Quema, and J. B. Stefani. Architecture-Based Autonomous Repair Management: An Application to J2EE Clusters. Proceedings of 24th IEEE Symposium on Reliable Distributed Systems (**SRDS**), Orlando, Florida, October 2005.

[BRUN04] E. Bruneton, V. Quéma, T. Coupaye, M. Leclercq, and J.B. Stefani. An Open Component Model and its Support in Java. Proceedings 7th International Symposium on Component-Based Software Engineering (CBSE 2004), Springer LNCS 3054, 2004.

[CAND 04] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. A Microrebootable System: Design, Implementation, and Evaluation. In Proceedings OSDI '04, 2004.

[CART05] Bruno Carton and Valentin Mesaros. Improving the Scalability of Logarithmic-Degree DHT-based Peer-to-Peer Networks. Euro-Par 2004, Pisa, Italy, August-September 2004.

[CHUN03] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. In *ACM SIGCOMM Comp. Comm. Review*, 33(3), 2003.

[CTM04] P. Van Roy and S. Haridi. Concepts, Techniques, and Models of Computer Programming. MIT Press, 929 pages, March 2004.

[CURRY] The Functional Logic Language Curry. http://www.informatik.uni-kiel.de/~mh/curry/

[DARWIN] The Darwin Project, Imperial College. http://www-dse.doc.ic.ac.uk/

[DKS05] Distributed K-Ary System: A Peer-to-Peer Middleware. http://dks.sics.se/, 2005.

[DMTF] Distributed Management Task Force. http://www.dmtf.org//

[ELAN03] S. El-Ansary, L. Onana Alima, P. Brand, and S. Haridi. Efficient Broadcast in Structured P2P Networks. In 2nd International Workshop on Peer-to-Peer Systems (IPTPS 03*)*, Berkeley, CA, Feb. 2003.

[ERLANG] Open Source Erlang, http://www.erlang.org/

[FLEU03] M. Fleury and F. Reverbel. The Jboss Extensible Server. Middleware 2003, Springer LNCS 2672, 2003.

[FLEX05] FlexiNET: Flexible Network Architecture for Enhanced Access Network Services and Applications. IST Summit, Dresden, June 2005.

[GANE03] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. IBM Systems Journal, 42(1), 2003.

[GARL00] D. Garlan and R. Monroe and D. Wile. Acme: Architectural Description of Component-Based Systems. Chapter 3 in [LEAV00].

[GHOD05] A. Ghodsi, L. Onana Alima, and S. Haridi. Low-Bandwidth Topology Maintenance for Robustness in Structured Overlay Networks. In *38th HICSS Conference*, Best Paper Award in Software Track, Hawaii, Jan. 2005.

[GHVR05] Ali Ghodsi, Seif Haridi, Peter Van Roy, Thomas Sjöland, Peter Sewell, Luc Onana Alima, Per Brand, Kevin Glynn, Rachid Guerraoui, and James Leifer. Building Internet-scale Distributed Systems Using Self-managing Overlays. Submitted to IEEE Computer, July 2005.

[GUMM03] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. ACM SIGCOMM'03.

[HASKELL] Haskell: A Purely Functional Programming Language. http://www.haskell.org/

[HERR05] Klaus Herrmann, Gero Mühl, and Kurt Geihs. Self Management: The Solution to Complexity or Just Another Problem? IEEE Distributed Systems Online, 6(1), Jan. 2005.

[IBM] Autonomic computing: IBM's perspective on the state of information technology. http://researchweb.watson.ibm.com/autonomic/

[KAAS03] F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In 2nd International Workshop on Peer-to-Peer Systems (IPTPS 03), Berkeley, CA, Feb. 2003.

[KINESTHETICS] Columbia University Programming Systems Lab, Kinesthetics eXtreme, http://www.psl.cs.columbia.edu/kx/

[KON00] F. Kon and M. Roman and P. Liu Mao and T. Yamane and L.C. Magalhaes and R. Campbell. Monitoring, Security and Dynamic Configuration with the dynamicTAO Reflective ORB. Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000), 2000.

[LEAV00] G. Leavens and M. Sitaraman (eds). Foundations of Component-Based Systems. Cambridge University Press, 2000.

[LUCK95] D. Luckham and J. Vera. An Event-based Architecture Definition Language. IEEE Transactions on Software Engineering, 21(9), 1995.

[MAGE95] J. Magee and N. Dulay and S. Eisenbach and J. Kramer. Specifying Distributed Software Architectures. Proceedings 5th European Software Engineering Conference, Springer LNCS 989, 1995.

[MEDV99] N. Medvidovic and D. S. Rosenblum and R. N. Taylor. A Language and Environment for Architecture-Based Software Development and Evolution. Proceedings of the 21st International Conference on Software Engineering (ICSE'99), 1999.

[MESA05] Valentin Mesaros, Raphaël Collet, Kevin Glynn, and Peter Van Roy. A Transactional System for Structured Overlay Networks. Research Report RR2005-01, Department of Computing Science and Engineering, Université catholique de Louvain, March 2005.

[MIS04] A. Mislove and P. Druschel. Providing administrative control and autonomy in peer-to-peer overlays. 3rd International Workshop on Peer-to-Peer Systems, San Diego, CA, Feb. 2004.

[MONT05] Alberto Montresor and Mark Jelasity and Ozalp Babaoglu. Chord on Demand. 5th International Conference on Peer-to-Peer Computing, Aug. 2005.

[MOZART] Mozart Programming System release 1.3.1. http://www.mozart-oz.org/ , July 2004.

[MOZ04] Multiparadigm Programming in Mozart/Oz. Second International Conference (MOZ 2004), Springer LNCS volume 3389, Charleroi, Belgium, Oct. 2004.

[OBJE05] The ObjectWeb Open Source Middleware Consortium. http://www.objectweb.org/.

[OCEANO] The Oceano Project. http://www.research.ibm.com/oceanoproject/

[OCEANSTORE] The OceanStore Project. http://oceanstore.cs.berkeley.edu/

[P2PK05] P2PKit: A Services Based Architecture for Deploying Robust Peer-to-Peer Applications. http://p2pkit.info.ucl.ac.be/, 2005.

[P2PS05] P2PS: A Peer-to-Peer Networking Library for Mozart/Oz, http://p2ps.info.ucl.ac.be/, 2005.

[ROC] The Recovery-Oriented Computing Project. http://roc.cs.berkeley.edu/

[ROWS01] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. Springer LNCS 2218, 2001.

[SARD05] Sardes: System Architecture for Reflective Distributed EnvironmentS, http://sardes.inrialpes.fr/, 2005.

[SCHM 04] A. Schmitt and J.-B. Stefani. The Kell Calculus: A Family of Higher-Order Distributed Process Calculi. In Global Computing, LNCS 3267, Springer, 2004.

[SCHU04] T. Schütt, A. Merzky, A. Hutanu, F. Schintke. Remote Partial File Access Using Compact Pattern Descriptions. IEEE/ACM Intl. Symp. on Cluster Computing and the Grid (CCGrid2004), pp. 1-8, April 2004.

[SCHU05] T. Schütt, F. Schintke, A. Reinefeld. Chord#: Structured Overlay Network for Non-Uniform Load Distribution. Technical Report ZR-05-40, August 2005.

[SHAK05] Ayman Shaker and Douglas S. Reeves. Self-Stabilizing Structured Ring Topology P2P Systems. Peer-to-Peer Computing 2005.

[SIT02] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables, 1st International Workshop on Peer-to-Peer Systems, 2002.

[SMARTFROG] The Smartfrog project. http://www.hpl.hp.com/research/smartfrog/

[SZYP02] Clemens Szyperski, with Dominik Gruntz and Stephan Murer. Component Software–Beyond Object-Oriented Programming. Addison-Wesley/ACM Press, Second Edition, 2002.

[STOI01] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, pages 149-160, San Diego, CA, Aug. 2001.

[SUF05] Sufatrio and R.H.C. Yap. Improving Host-Based IDS with Argument Abstraction to Prevent Mimicry Attacks. 8th International Symposium on Recent Advances in Intrusion Detection, 2005, 146-164.

[SWAN] SWAN: Self-aWare mAnagemeNt. French RNRT exploratory project, http://swan.elibel.tm.fr/

[WERM01] M. Wermelinger and A. Lopes and J. Fiadeiro. A Graph Based Architectural (Re)configuration Language. Proceedings ESEC/FSE `01, V. Gruhn (ed), ACM Press, 2001.

[WU05] Y. Wu and R.H.C. Yap. A User-level Framework for Auditing and Monitoring. 21st Annual Computer Security Applications Conference, 2005, pages 84-94.

[ZHAO04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications, 22(1), Jan. 2004.